1981

Development of improved finite elements formulation for shallow water equations.

Woodard, Edward T.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

DEVELOPMENT OF IMPROVED
FINITE ELEMENT FORMULATION FOR
SHALLOW WATER EQUATIONS

by

Edward T. Woodward

September 1981

Thesis Advisors: R.T. Williams
U.R. Kodres

Approved for public release; distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Development of Improved Finite Element Formulation for Shallow Water Equations		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis September 1981
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Edward T. Woodward		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE September 1981
		13. NUMBER OF PAGES 170
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Finite elements
Shallow water equations
Voriticity-divergence formulation
Semi-implicit
piecewise linear basis function
piecewise constant basis function
primitive barotropic equations
Galerkin method
numerical prediction model
variable size elements

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The basic principles of the Galerkin finite element method are discussed and applied to two different formulations; one using different basis functions and the other using the vorticity-divergence form of the shallow water equations. Each formulation is compared to the primitive form of the equations developed by Kelley (1976). The testing involves a comparison of three finite element prediction models using variable size elements. Equilateral elements significantly improve the solution and are used in most of the comparisons.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The formulation using different basis functions produces poorer results than the primitive formulation. The vorticity-divergence formulation produces superior results while executing faster than the primitive model. However, it does require more storage and the relaxation parameters are sensitive to the domain geometry. The computer implementation for the vorticity-divergence model is discussed and the source listing is included.

Approved for public release; distribution unlimited.

DEVELOPMENT OF IMPROVED FINITE ELEMENT FORMULATION FOR SHALLOW-WATER EQUATIONS

by

Edward T. Woodward
Captain, United States Air Force
B.A., University of Maine, 1970

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN COMPUTER SCIENCE
and
MASTER OF SCIENCE IN METEOROLOGY

from the

NAVAL POSTGRADUATE SCHOOL
September 1981

ABSTRACT

The basic principles of the Galerkin finite element method are discussed and applied to two different formulations; one using different basis functions and the other using the vorticity—divergence form of the shallow water equations. Each formulation is compared to the primitive form of the equations developed by Kelley (1976). The testing involves a comparison of three finite element prediction models using variable size elements. Equilateral elements significantly improve the solution and are used in most of the comparisons. The formulation using different basis functions produces poorer results than the primitive formulation. The vorticity-divergence formulation produces superior results while executing faster than the primitive model. However, it does require more storage and the relaxation parameters are sensitive to the domain geometry. The computer implementation for the vorticity-divergence model is discussed and the source listing is included.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

The author belives that most of the credit should go to Professor R.T. Williams for the education that he gave me through his patient teaching and guidance during the entire research and writing of this paper. Acknowledgement is also due Professor U.R. Kodres for the many useful suggestions, encouragement and reading the manuscript.

Special thanks to each department chairman, Professor G.H. Bradley, Computer Science, Professor R.J. Renard, Meteorology, and Professor G.J. Haltiner, former Meteorology chairman, for allowing me to pursue the two degrees and accepting this thesis as partial fulfillment for the requirements of both degrees.

Thanks are extended to Lt. R.G. Kelley whose research was the forerunner to this investigation and Capt. M. Older for the many useful comments he offered during the implementation. Particular thanks to Dr. M.J.P. Cullen whose personal suggestions provided the direction for this work.

I give my love to my wife, Joy, for her infinite patience, for being the first guinea pig in reading and preparing this manuscript. Finally, to my family, Kim, Mark and Joy thanks for enduring and encouraging me throughout my graduate studies at the Naval Postgraduate School.

I. INTRODUCTION

Shuman (1978) claims that progress in numerical modeling of the general circulation has been to some degree dictated in the past by the rate of development in the field of computer technology. However, the limited ability to parameterize the effects of small-scale processes in terms of large scale motions has been an equally important limiting factor. Essentially, the major problem of numerical modeling of the general circulation is simply that of producing a very long range numerical weather forecast.

Certainly the equations used in the models must be more sophisticated to include those physical processes which are unimportant for a short range forecast, but may become crucial as the length of the forecast is extended. Another area where concentrated efforts have improved the forecast involves the computational techniques employed to approximate and solve the governing equations of the models.

The motivation behind this thesis is to investigate the application of a relatively new computational technique to the field of numerical weather prediction. The finite element method, long established in engineering, has been seriously considered only during the past decade in meteorology. This method has great potential for application in atmospheric prediction models.

A. BACKGROUND

The most common numerical integration procedure for weather prediction has been the finite difference method in which the derivatives in the differential equations of motion are replaced by finite difference approximations at a discrete set of points in space and time. The resulting set of equations, with appropriate restrictions, can then be solved by algebraic methods. Until recently, the finite difference method has been the workhorse in atmospheric prediction models, from their first computer implementation to the present.

With the introduction of each new generation of computers, the gap between numerical forecasts and atmospheric observations has decreased. The rate at which this gap decreased has slowed down and appears to be leveling off. This would indicate that computer technology may not be the primary obstruction to better numerical forecasts. In fact, bigger and faster computers alone have demonstrated their inability to significantly improve the numerical forecast.

For example, a major limiting factor of finite difference approximations is the truncation error. The National Weather Service 7 Layer Primitive Equation Model (7LPE Model), operational from 1966 to 1980, had truncation errors which increased at a rate proportional to the square of the grid spacing. That is, the smaller the grid interval,

the smaller the truncation error. To increase its accuracy would require increasing the grid matrix density. This would require increased computer storage and computational time. State of the art computers are capable of providing these additional resources.

The problem now goes beyond numerical techniques and computer technology. Operationally, the National Weather service is not capable (due to monetary restrictions) of providing a denser concentration of atmospheric observations. Therefore, with the present density of initial data (observations) and objective analysis techniques (getting the data for grid points by interpolating from observed data sources), reducing the grid spacing further on the 7LPE Model does not significantly increase the accuracy of the solution.

This additional computer capability can not be utilized using finite difference methods. Therefore, new numerical integration techniques must be investigated, such that given the same density of observed data, superior solutions are produced.

Two alternative techniques, the spectral method and the finite element method, have started to gain attention. Both the spectral and finite element methods require more computational time per forecast time step than does the finite difference method. For example, the finite element method requires an equation solver to invert a larger matrix

at each time step for each variable. In this sense, these methods were held back by computer technology, but recent advances in computer technology (i.e. larger and faster storage devices, multi-processors, etc) have made these alternative numerical techniques competitive.

For long range weather predictions, the spectral method applied over the globe or hemisphere is a natural method, due to the existence of efficient transforms for the nonlinear terms on spherical geometry. It also eliminates the truncation error for the horizontal space derivatives and the nonlinear instability (aliasing). For these reasons, global spectral models have been developed and implemented on an operational level, replacing the global finite difference models.

However, because the spectral harmonics are globally rather than locally defined, it is thought that for problems of more detailed limited area forecasting, the finite element method is more suitable. Pioneering work to adapt finite element methods to meteorological applications has been done by Cullen (1973,1974 and 1979), Staniforth and Mitchell (1977), Hinsman (1975) and Kelley (1976). The most recent finite element meteorological model at the Naval Postgraduate School was written by Kelley (1976) with the collaboration of Dr. R.T. Williams. It is this study that will serve as a basis for this thesis. The model written by Kelley will be altered and used for comparative testing with

improved finite element forms implemented by this author. Some of the techniques and codes developed by Kelley are also employed in this thesis. Older (1981) developed a technique to smoothly vary the grid geometry in the domain. This technique is also implemented both on Kelley's model and with the new formulation to give greater versatility when testing the model performance.

3. OBJECTIVES

The objectives for this thesis can be divided into two categories: 1) meteorology, 2) computer science. First, the meteorological objectives of developing improved finite element forms for shallow water equations are as follows:

1) - Older (1981) after collaboration with Dr. M.J.P. Cullen, showed how equilaterally shaped elements produced significantly better results than did other triangular elements. Kelley (1976) used right triangular elements in the implementation of a two-dimensional finite element model using the primitive form of the shallow water equations. A considerable amount of small-scale noise was observed in the solution. Hereafter, this model, which was developed by Kelley (1976), will be referred to as the primitive model. This first objective involves re-implementing the primitive model using equilaterally shaped elements and comparing the results to those in Kelley's thesis.

2) - Williams and Zienkiewicz (1981) presented new finite element techniques for formulations for the shallow water equations, which use differently shaped functions to approximate the different dependent variables, which in effect stagger the variables. Schoenstadt (1980) demonstrated the advantage of spatial staggering of dependent variables in finite difference models. The application of this technique to finite element models is a natural extension, and excellent results were obtained by Williams and Zienkiewicz (1981) from application of these new formulations on linearized one dimensional cases. The objective here is to implement the new forms on the primitive model and again do quanitative comparisons of the results.

3) - The major emphasis in this study deals with the implementation and comparison of the vorticity divergence form of the shallow water equations, which is described in detail in Chapter III. This formulation has the following advantages. First, the geostrophic adjustment process is treated better than in the primitive form of the equations. Secondly, the velocity and height fields are evaluated at the same grid point, where the best primitive form requires staggering these dependent variables. And thirdly, a larger time step is allowed due to the semi implicit form of the equation. Again comparisons between the results from the vorticity divergence and primitive model are presented.

The computer science aspect of this thesis was primarily devoted to the implementation of the different models and the style and architecture of the program. Finite element methods require more computational time than do finite difference methods, not only in the solution of the equations, but also in the amount of computation required to evaluate each term in the equations.

The implementations of these two dimensional models, although complex when viewed from the surface, have a lot of generality and redundancy in the operations required. Versatile modules can be written to ease the implementation and facilitate changes. The objective here is to efficiently implement these new forms and demonstrate the utility of these versatile modules for future implementations.

C. THESIS STRUCTURE

This thesis presents the results obtained from tests of the various finite element formulations. The results are compared to those from the primitive model written by Kelley (1976). Accompanying the results is a detailed discussion of the reformulation and implementation process.

Chapter II of the thesis presents a tutorial of the finite element method and the area coordinates system used in the evaluation of the element integration. The Galerkin finite element method used in all the models is developed and applied to the advection equation in one dimension.

Chapter III presents the detailed description of the vorticity-divergence shallow water model. Here the equations are shown and written using the Galerkin method. A discussion of the computational technique used is presented along with the model's physical parameters.

Chapter IV presents a descriptive overview of the computer implementation. The chapter includes a list of options available for testing, a brief description of the matrix compaction technique and the formulations of the versatile modules used to implement the complex equations.

Chapters V through VII discuss the results obtained from the different experiments. Chapter V briefly describes the primitive model used for all comparisons and the results from changing the element shape to equilateral triangles. Chapter VI reformulates the primitive model so that the geopotential is staggered with respect to the velocity variable. For simplicity, the continuity equation is also linearized. Chapter VII compares the results from the vorticity-divergence model developed in Chapter III to those from the primitive model.

The last chapter summarizes the results from all the experiments and identifies what areas require follow on work. The source code for the vorticity-divergence model is presented in Appendix A.

II. FINITE ELEMENTS

As is often the case with an original development, it is rather difficult to quote an exact date on which the finite element method was invented, but the roots of the method can be traced back to these separate groups: applied mathematicians, physicists and engineers. Since the early developments of the finite element method, a large amount of research has been devoted to the technique. However, the finite element method obtained its real impetus by the independent developments carried out by engineers. Its essential simplicity in both physical interpretation and mathematical form has undoubtedly been as much behind its popularity as is the digital computer which today permits a realistic solution of even the most complex situations.

The name " finite element " was coined in a paper by R.W. Clough, in which the technique was presented for plane stress analysis, as discussed in Bathe (1976). While finite element methods have made a deep impact via the field of solid mechanics, where it can be said that today they represent the generally accepted method of discretizing continuum problems for computer-based solution, the same appears not to be true in fluid mechanics or atmospheric prediction.

Numerous finite element formulations are currently available. Strang (1973), Norrie (1973) and Zienkiewicz (1971) present detailed theoretical discussions of each. The Galerkin method, the most popular finite element method, is described in detail below and used in the equation formulation later.

A. FINITE ELEMENT CONCEPT

The problem of solving partial differential equations can be specified in one of two ways. In the first, finite difference methods specify the dependent variables at certain grid points in space and time, and the derivatives are evaluated using Taylor series approximations. Secondly, the calculus of variation requires the minimization of a functional over a domain, where a functional is defined as a variational integral over the domain.

The calculus of variation approach creates a purely physical model where the functional equivalent to the known differential equations are known. Its major disadvantage is that it limits the method only to those problems for which functionals exist. Finite element methods, an extention of this method, derive mathematical approximations directly from the differential equations governing the problem. The advantage here is that it extends the method to a range of problems for which a functional may not exist, or has not been discovered.

The finite element method divides the domain into subdomains or finite elements (usually of the same form). Nodes are located along the boundary of the elements, usually at the element vertices and at strategic positions (midside, centroid, etc.) in the interior and on the sides of faces of elements.

Commonly used elements are triangular, polygonal or polyhedral in form for two-dimensional problems. The choice of elements depends on the type of problem, the number of elements desired, the accuracy required and the available computing time. To begin with, the element must be able to represent derivatives of up to the order required in the solution procedure, and to guarantee continuous first derivatives across the element boundaries to avoid singularities. Triangular elements are employed in this thesis because they can be used effectively to represent irregular boundaries, and/or geometry, and also to concentrate coordinate functions in those regions of the domain where rapidly varying solutions are anticipated.

Consider the problem of solving approximately the differential equation

$$L(u) = f(x) \qquad \text{II-1}$$

where L is a differential operator, u the dependent variable, and $f(x)$ is a specified forcing function. Suppose that II-1 is to be solved in the domain $a \leq x \leq b$ and that

appropriate boundary conditions are provided. The residual R is formed from II-1 as follows:

$$L(u) - f(x) = R \qquad \text{II-2}$$

The critical step is to select a trial family of approximate solutions (the members of a trial family are often called basis functions). The basis function is prescribed (functionally) over the domain in a piecewise fashion, element by element, and are generally a combination of low order polynominals. A one dimensional example is shown in Figure 1, wherein the domain (x axis) is divided into six elements (line segments) A through F. The basis functions are linear and one is shown for node 4 only in Figure 1. The function has a value of unity over node 4, and decreases linearly to is zero at nodes 3 and 5 and zero elsewhere.

Consider a series of linearly independent basis functions $V_j(x)$, as in Figure 1. Now $u(x)$ can be approximated with a finite series as follows:

$$u(x) = \sum_j \phi_j V_j(x) = \phi_j V_j \qquad \text{II-3}$$

where ϕ_j is the coefficient of the jth basis function and has a value equal to u at node j.

Substituting this approximate solution II-3 wherever u appears in the differential equation II-1

$$L(\phi_j V_j) - f(x) = R \qquad \text{II-4}$$

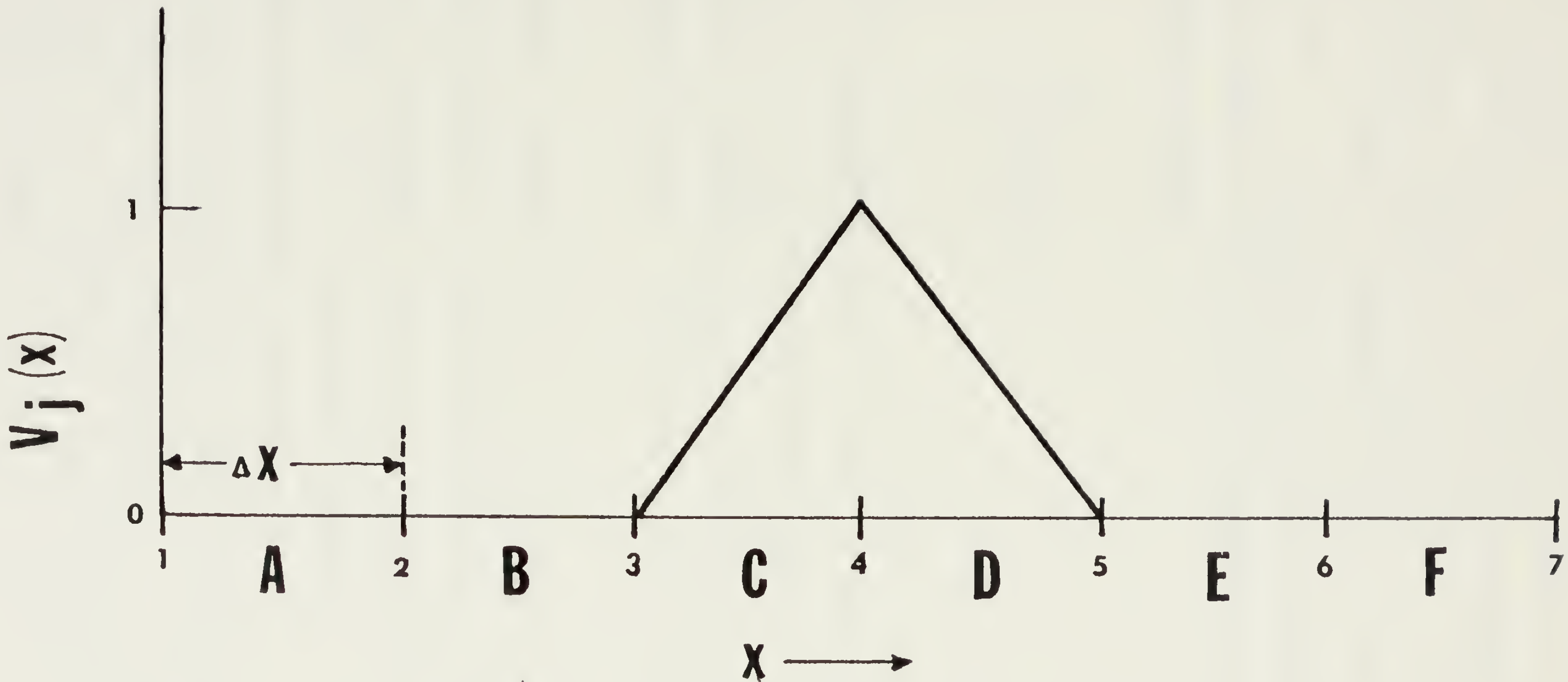


Figure 1. Piecewise linear basis function.

The best solution will be one which in some sense reduces the residual R to a minimum value at all points in the domain. By definition, the residual obtained using the exact differential equation is identically zero everywhere. The residual R, formed in equation II-4, is minimizied when multiplied with a weighting function, integrated over the domain and set equal to zero. This process is known as the weighted residual method

$$\int_a^b RW\,dx = 0 \qquad \text{II-5}$$

where W is the weighting function and is referred to as the test function in the following development. The weighted residual method minimizes the errors of the residuals, such that the summation of all the positive and negative errors add to zero.

The Galerkin method, the most popular finite element method, is more general in application and is a special case of the method of weighted residuals, as discussed by Pinder and Gray (1977). The requirement imposed on the weighted residual method forming the Galerkin method is:

* the test (weighting) function be equal to the basis (trial) function W = V . This process leads in general to the best approximation of the solution.

The final Galerkin form is obtained by substituting II-4 into II-5, yielding

$$\int_a^b W_i L(\phi_j V_j)dx - \int_a^b W_i f(x)dx = 0 \qquad \text{II-6}$$

If this procedure is repeated for N points in the domain a system with N equations and N unknowns will be generated.

3. GALERKIN APPLICATION

The following example taken in part from Haltiner and Williams (1980) applies the Galerkin method to the advection equation with linear elements

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \qquad \text{II-7}$$

This equation is dependent in both time and space. The treatment of time variation is important for most meteorological prediction problems. The Galerkin method is not applied to the time dependence because it is more convenient to use finite differences in time,as is done with this example later. The same treatment is applied to the prognostic equations later, where two finite differencing methods are employed to do the time integration.

The Galerkin procedure represents the dependent variable u(x.t) with a sum of functions that have the prescribed

spatial structure as in Figure 1. Approximate $u(x,t)$ with the finite series as follows

$$u(x,t) = \sum_{j=1}^{N} \phi_j(t)V_j(x) = \phi_j V_j \qquad \text{II-8}$$

where the coefficient $\phi_j(t)$, a function of time, is the scalar value of u at node j. The basis functions, $V_j(x)$, are functions of space only and j equals 1 to 7 for the example in Figure 1. The repeated subscript in this form implies a sum over the repeated subscript.

The Galerkin equation for the advection equation II-7 is obtained by setting $L = c(\partial()/\partial x)$ and substituting in the approximate solution II-8 wherever u is found.

$$\sum_{j=1}^{N} \frac{\partial \phi_j}{\partial t} \int_a^b V_j V_i \, dx + c \sum_{j=1}^{N} \phi_j \int_a^b \frac{\partial V_j}{\partial x} V_i = 0 \qquad \text{II-9}$$

where $i = 1$ to N, V_i the test function and V_j the basis function. The domain of integration is given by $a \leq x \leq b$, and the integration is done in a piecewise fashion, element by element.

In this one-dimensional case, an equation like II-9 is written for each node, i. Considering node 4, what are the possible non-zero contributions from equation II-9? Figure 2 illustrates the basis and the test function interaction during the piecewise integration process. From the

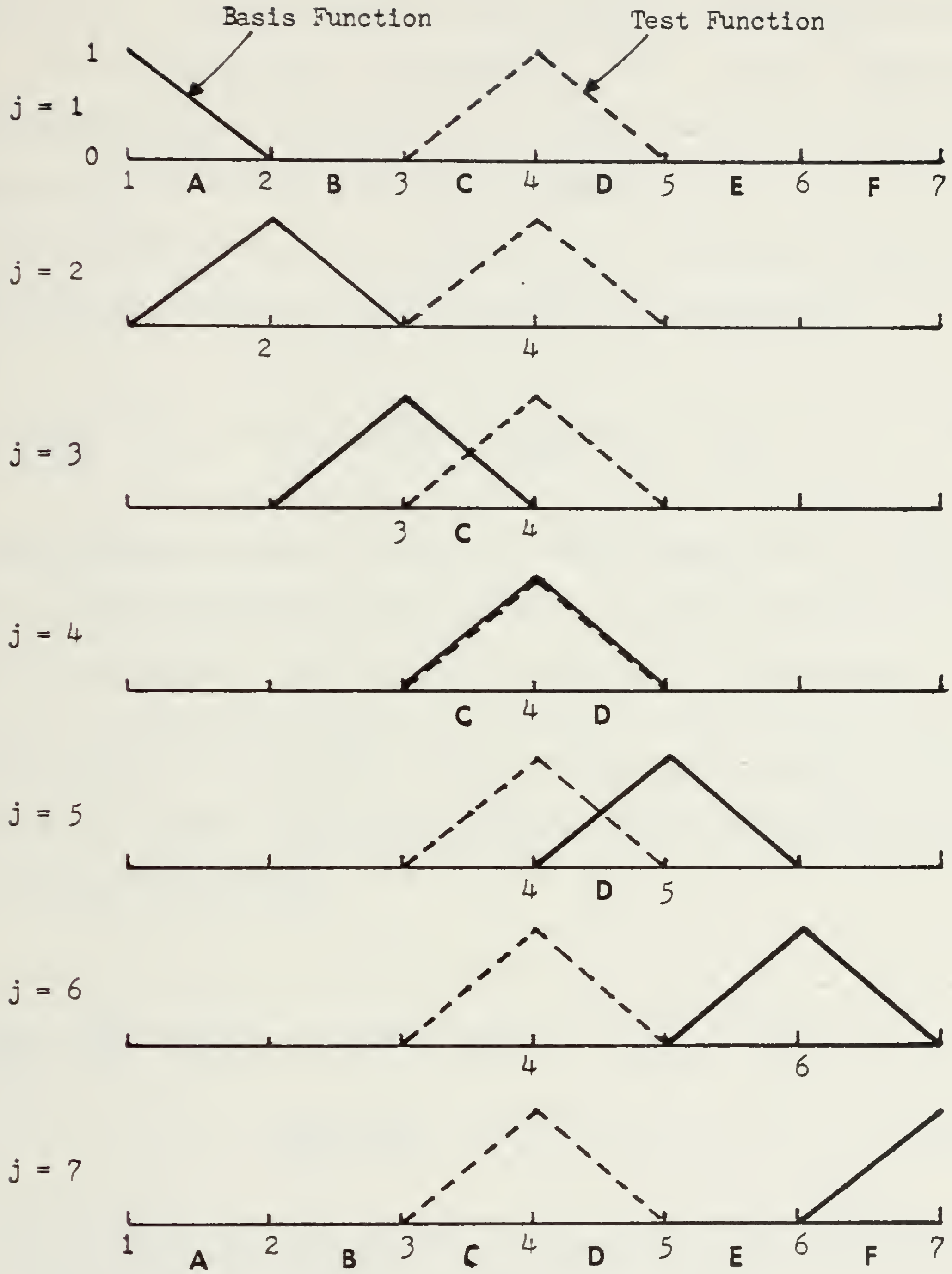


Figure 2. Basis and test function interaction during the piecewise integration process.

definition of the basis and the test function, locally defined as unity at node j and linearly decreasing to zero at $j \pm 1$ and zero elsewhere, the only non—zero contributions are made when $j = 3$ over element C, $j = 4$ over elements C and D, and $j = 5$ over element D.

The evaluation of II-9 for $i = m$, which is given in Haltiner and Williams (1981), leads to the equation:

$$\frac{1}{6}\frac{d}{dt}(u_{m+1} + 4u_m + u_{m-1}) + \frac{c}{2\Delta x}(u_{m+1} - u_{m-1}) = 0 \qquad \text{II-10}$$

The boundary points, which in this example are nodes 1 and 7, are evaluated in the same way as the interior nodes, with the exception that cyclic conditions are imposed.

The time discretization of II-10 is done using a finite difference scheme. Applying leapfrog time differencing gives the following equation

$$\frac{1}{12\Delta t}(u^{n+1}_{m+1} - u^{n-1}_{m+1} + 4(u^{n+1}_{m} - u^{n-1}_{m}) + u^{n+1}_{m-1} - u^{n-1}_{m-1})$$
$$+ \frac{c}{2\Delta x}(u^{n}_{m+1} - u^{n}_{m}) = 0 \qquad \text{II-11}$$

The resultant equation set, in matrix form, contains an NxN matrix where N is the number of nodes.

The transition from one-dimension to two is mathematically identical. The domain is now subdivided into finite areas, which are triangles in this implementation and

the basis functions are linear. However, now they are pyramid shaped with value unity at the center and decrease to zero at the surrounding nodes, and are zero elsewhere. Figure 3 shows this basis function for node 28 outlined in heavy black. The value at any node again can be approximated by II-3, where j ranges over all nodes connected to node i including i itself. The connectivity for node i = 28 in Figure 3 is j = 15,16,27,28,29,39 and 40.

The integration is still over the entire domain. With both the basis and the test function zero over the domain, except locally over each element, the global integration can be performed by integrating locally over each element. By definition, this integration can be expressed as an inner product of both functions (i.e. basis, test) as follows:

$$\langle V_j, V_i \rangle = \iint_A V_j V_i \, dA \qquad \text{II-12}$$

Using this definition and the repeated subscript notation equation II-9 becomes

$$\dot{\phi}_j \langle V_j, V_i \rangle + c\, \phi_j \langle V_{jx}, V_i \rangle = 0 \qquad \text{II-13}$$

where the dot implies differentiation with respect to time, and the second subscript implies differentiation with respect to the second subscript. The local integration may be calculated directly from exact expressions derived from area coordinates described in detail in the next subsection.

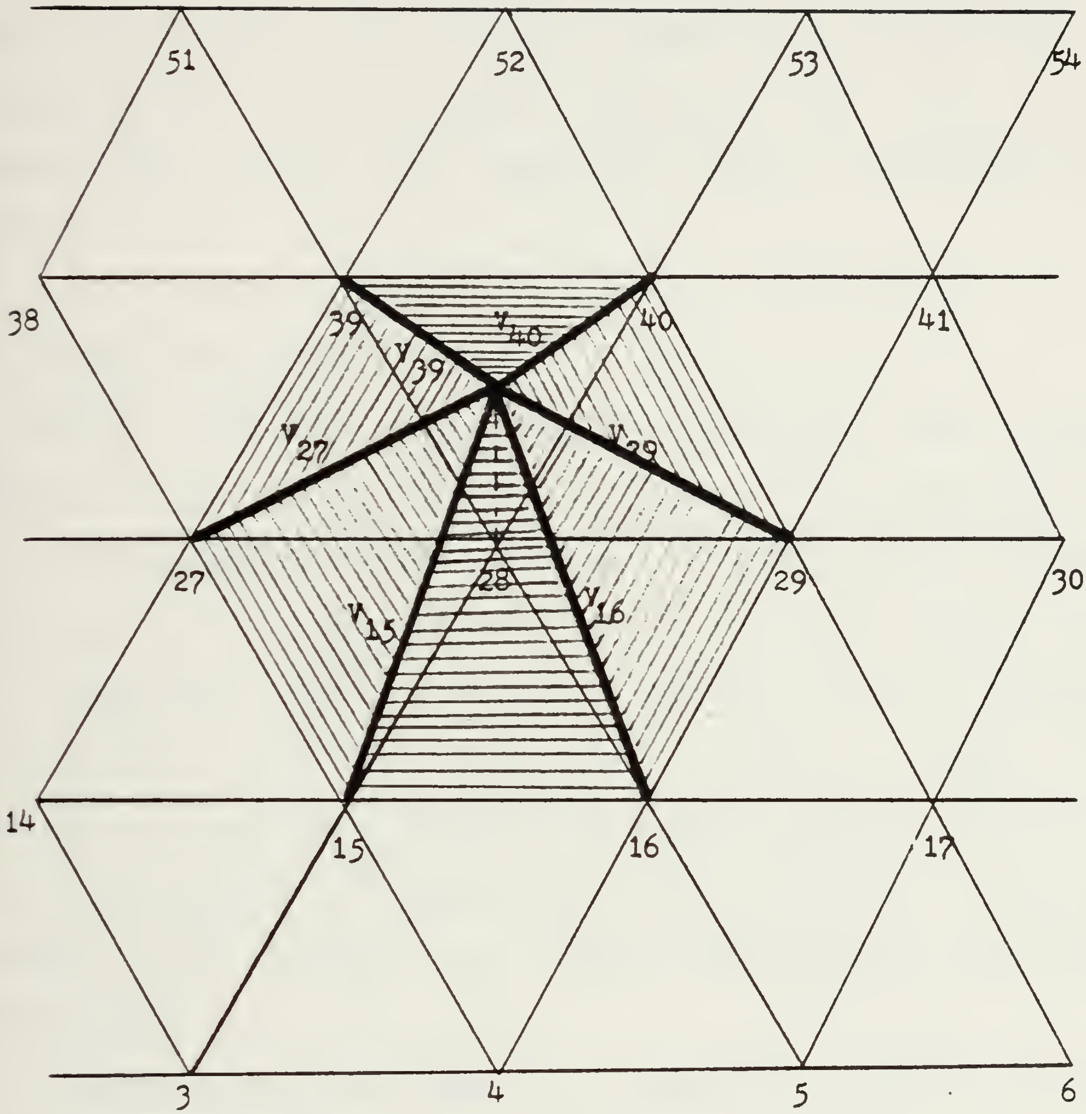


Figure 3. Basis function for node 28. The shaded area is the complete basis function and the V_j , where j = 15, 16,27,28,29,39,40 are jth node basis functions for node 28. The dashed line at node 28 has length unity.

In summary, the Galerkin procedure involves subdividing the domain into finite elements, approximating the dependent variables by a linear combination of low order polynomials and substituting them into the equations. The equation is multiplied by a test function, integrated over the entire domain and finally the resulting system of equations is solved.

C. AREA COORDINATES

While the Cartesian coordinate system is the natural choice of coordinates for most two dimensional problems, it is not convenient when working with triangularly shaped elements. It is therefore necessary to define a special set of normalized coordinates for a triangle. Area, or natural coordinates as they are commonly called, reduce the formidable task of integrating products between the basis and test functions and their derivatives over a triangular element and result in easily computable and exact expressions.

The following development is taken in part from the formulation by Zienkiewics (1971). Consider the triangular element illustrated in Figure 4. There is a one-to-one correspondence between the Cartesian coordinates (X,Y) and the area coordinates (L_1, L_2, L_3) for the element. Let A denote the area of the triangle and A_1, A_2 and A_3 the areas of the subtriangles in Figure 4 such that $A = A_1 + A_2 + A_3$.

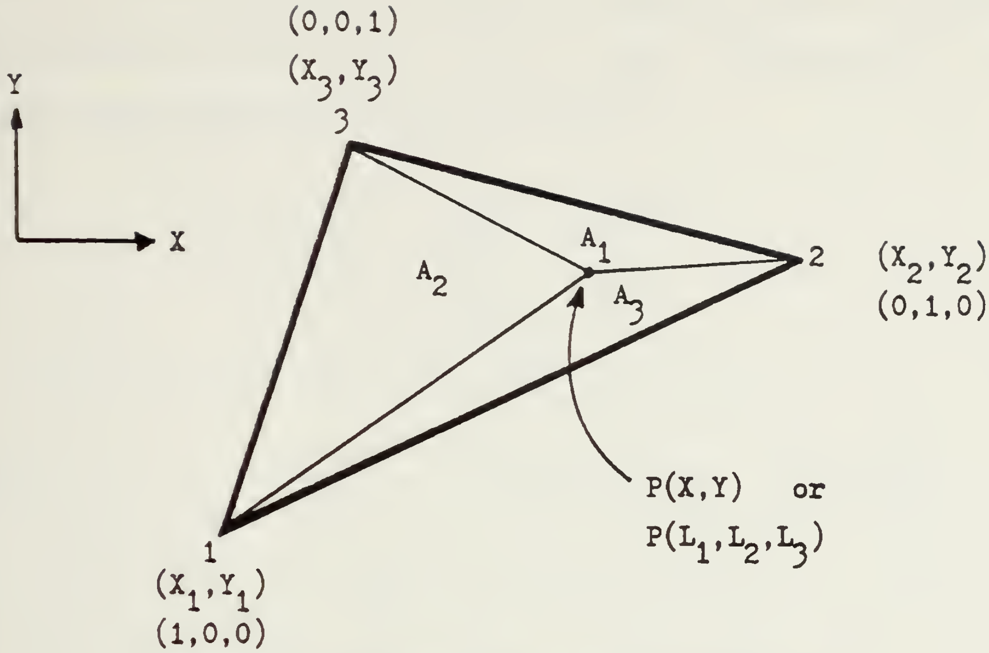


Fig 4. Cartesian vs. area coordinates

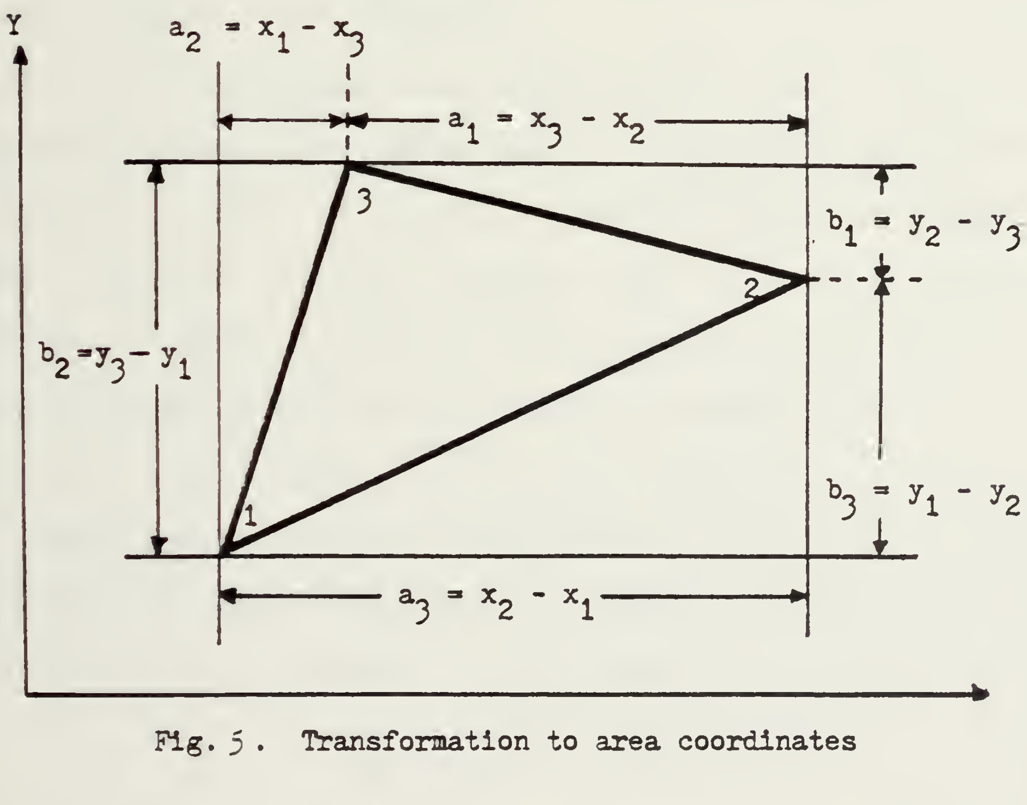


Fig. 5. Transformation to area coordinates

The relationship between a point P(X,Y) in Cartesian coordinates and $P(L_1, L_2, L_3)$ in area coordinates can be seen by the following transformations

$$X = L_1X + L_2X + L_3X$$
$$Y = L_1Y + L_2Y + L_3Y \qquad \text{II-14}$$
$$1 = L_1 + L_2 + L_3$$

where $L_1 = \frac{A_1}{A}$, $L_2 = \frac{A_2}{A}$ and $L_3 = \frac{A_3}{A}$

and

$$L_1 = (2A + b_1X + a_1Y)\ /2A$$
$$L_2 = (2A + b_2X + a_2Y)\ /2A \qquad \text{II-15}$$
$$L_3 = (2A + b_3X + a_3Y)\ /2A$$

where 2A is twice the area of the triangle and the a´s and b´s are defined as in Figure 5.

It is worth noting that every tuple (L_1, L_2, L_3) corresponds to a unique pair (X,Y) of Cartesian coordinates. In Figure 4, $L_1 = 1$ at vertex 1 and Ø at vertices 2 and 3. A linear relation exists between the area and Cartesian coordinates which implies that values for L_1 vary linearly over the triangle with a value one at vertex 1 and a value of zero at vertices 2 and 3; and similarly for L_2 and L_3. This demonstrates how each component in the tuple (L_1, L_2, L_3) behaves over the triangle as do the linear basis and test functions over the element, as was seen in Figure 4. Clearly

$$L_i = V_i \qquad \text{II-16}$$

where V_i is a linear function of the Cartesian coordinates (i.e. basis, test).

Zienkiewicz (1971) shows that it is possible to integrate any polynomial in area coordinates using the simple relationship

$$\iint_A L_1^m L_2^n L_3^p \, dxdy = \frac{m!\ n!\ p!}{(m + n + p + 2)!} 2A \qquad \text{II-17}$$

where m, n and p are positive integers and A is the elementary area. For an example of this integration technique using inner product notation, equation II-12 is evaluated as follows

$$\langle V_j \cdot V_i \rangle = \begin{cases} \displaystyle\iint_A V_i^2 \, dxdy = \frac{2!\ 0!\ 0!}{(2 + 0 + 0 + 2)!} 2A = \frac{A}{6} & i = j \\ \displaystyle\iint_A V_j V_i \, dxdy = \frac{1!\ 1!\ 0!}{(1 + 1 + 0 + 2)!} 2A = \frac{A}{12} & i \neq j \end{cases} \qquad \text{II-18}$$

The differential operations in area coordinates follow directly from the differentiation of (II-15) where

$$\frac{\partial}{\partial x} = \sum_{i=1}^{3} \frac{b_i}{2A} \frac{\partial}{\partial L_i} \qquad \text{II-19}$$

and

$$\frac{\partial}{\partial y} = \sum_{i=1}^{3} \frac{a_i}{2A} \frac{\partial}{\partial L_i} \qquad \text{II-20}$$

As explained earlier (see Equation II-16), V_i is a linear function (i.e. basis, test) which equals a component L_i of the area coordinate tuple. Therefore

$$\frac{\partial V_j}{\partial L_i} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \qquad \text{II-21}$$

Consequently $\partial V_j / \partial x$ for $j = 1$ is

$$V_{jx} = \frac{\partial V_j}{\partial x} = \frac{b_1}{2A}\frac{\partial V_1}{\partial L_1} + \frac{b_2}{2A}\underbrace{\cancel{\frac{\partial V_1}{\partial L_2}}}_{0} + \frac{b_3}{2A}\underbrace{\cancel{\frac{\partial V_1}{\partial L_3}}}_{0} = \frac{b_1}{2A} \qquad \text{II-22}$$

As an example, consider the inner product $\langle V_{jx}, V_i \rangle$ at vertices $j = 2$, $i = 1$. This integration is evaluated as

$$\langle V_{2x}, V_i \rangle = \iint_A \frac{b_2}{2A} V_1 \, dxdy \qquad \text{II-23}$$

$$= \frac{b_2}{2A} \frac{1!\ 0!\ 0!}{(1 + 0 + 0 + 2)!} 2A = \frac{b_2}{6}$$

Therefore any inner product in the formulation can be readily evaluated using area coordinates. Another benefit of using this coordinate system is that all of the inner products are functions of space only and need be computed only once.

III. SHALLOW WATER MODEL

The governing equations for this model are derived by making several simplifying assumptions on the primitive equations of motion, which then give the barotropic shallow water equations. However, as mentioned previously, the shallow water equations describe many significant features of the large-scale motion of the atmosphere, and therefore have been used in numerous experiments over the years.

The vorticity-divergence form of the equations has several advantages. Williams (1981) has shown that the geostrophic adjustment process is treated much better with the vorticity divergence formulation than with a direct treatment of the primitive form of the shallow water equations, such as was used by Kelley (1976). This formulation also allows the velocity components and the height to be carried at the same nodal points, whereas the best scheme for the primitive form of the equations requires staggering of the fields, as seen in Schoenstadt (1980). The vorticity divergence form of the equations is also convenient for the application of semi-implicit differencing, which saves considerable computer time.

A. GOVERNING EQUATIONS

The primitive form of the shallow water equations in Cartesian coordinates is

$$\frac{\partial \phi}{\partial t} + D\bar{\Phi} = -\frac{\partial}{\partial x}(\phi u) - \frac{\partial}{\partial y}(\phi v) \qquad \text{III-1}$$

$$\frac{\partial u}{\partial t} = -\frac{\partial \phi}{\partial x} + Qv - \frac{\partial K}{\partial x} \qquad \text{III-2}$$

$$\frac{\partial v}{\partial t} = -\frac{\partial \phi}{\partial y} - Qu - \frac{\partial K}{\partial y} \qquad \text{III-3}$$

Equation (III-1) is the continuity equation and the III-2 and III-3 are the momentum equations, respectively. The variables are defined as follows:

x,y - the spatial coordinates of the domain

u,v - components of the wind vector

ϕ geopotential = (gravity x free surface height)

$\bar{\Phi}$ - mean geopotential = 49,000 meters^2/seconds^2

t time

K - kinetic energy

Q - absolute vorticity = $(\zeta + f_0)$

ζ relative vorticity

f_0 - coriolis force (mid-channel f-plane)

D divergence

The shallow water equations can be written in vorticity divergence form as follows:

$$\frac{\partial \phi}{\partial t} + D\phi = -\frac{\partial}{\partial x}(\phi u) - \frac{\partial}{\partial y}(\phi v) \qquad \text{III-4}$$

$$\frac{\partial S}{\partial t} = -\frac{\partial}{\partial x}(uQ) - \frac{\partial}{\partial y}(vQ) \qquad \text{III-5}$$

$$\frac{\partial D}{\partial t} + \nabla^2 \phi = -\frac{\partial}{\partial x}(vQ) - \frac{\partial}{\partial y}(uQ) - \nabla^2 K \qquad \text{III-6}$$

where III-4 is the same continuity equation as III-1, III-5 is the vorticity equation and III-6 is the divergence equation.

Because of the vorticity divergence form of the equations, it becomes necessary to solve the time dependent variables ζ and D in terms of Ψ, the stream function (rotational part of the wind), and χ, the velocity potential (divergent part of the wind). The initial fields for the model will be in terms of Ψ, χ and ϕ.

The following diagnostic relationships are defined and used later in the solution of the equation set.

$$u = -\Psi_y + \chi_x, \qquad \text{III-7}$$

$$v = \Psi_x + \chi_y, \qquad \text{III-8}$$

where the subscript implies differentiation,

$$K = \frac{u^2 + v^2}{2} \quad \text{kinetic energy}, \qquad \text{III-9}$$

$$uQ = u(\zeta + f_o), \qquad \text{III-10}$$

$$vQ = v(\zeta + f_o), \qquad \text{III-11}$$

$$\alpha = \phi u, \qquad \text{III 12}$$

$$\beta = \phi v, \qquad \text{III-13}$$

$$\zeta = \nabla^2 \Psi, \qquad \text{III-14}$$

$$D = \nabla^2 \chi. \qquad \text{III-15}$$

3. EQUATION FORMULATION

The Galerkin method described in Chapter II is now applied to equations III-4 through III-15. For ease of comprehension, the shorthand inner product notation as in II-12 will be used to simplify the equations. The detailed Galerkin formulation will be shown for equation III-7, the u component of motion. The method follows directly from the example in Chapter II of this thesis, which in turn follows in part from Kelley (1976) and Baltiner and Williams (1981).

Consider equation III-7 and assume that each variable u, Ψ and χ is approximated by

$$\begin{aligned} u &= u_j V_j , \\ \Psi &= \Psi_j V_j , \\ \chi &= \chi_j V_j , \end{aligned} \qquad \text{III-16}$$

where the repeated subscripts indicate summation over the range of the subscript. Substituting these approximate solutions into III-7 yields

$$u_j V_j = \frac{\partial}{\partial y}(\Psi_j V_j) + \frac{\partial}{\partial x}(\chi_j V_j) \qquad \text{III-17}$$

Since only the basis function V_j is a function of space, III-17 may be further simplified by factoring out the time dependent coefficients.

The next step requires multiplying by a test function V_i as discussed in Chapter II, and integrating over the area domain

$$u_j \iint_A V_j V_i \, dA = - \Psi_j \iint_A \frac{\partial V_j}{\partial y} V_i \, dA$$

$$+ \chi_j \iint_A \frac{\partial V_j}{\partial x} V_i \, dA \qquad \text{III-18}$$

The final form in inner product notation is

$$\langle u_j V_j, V_i \rangle = - \langle \Psi_j V_{jy}, V_i \rangle + \langle \chi_j V_{jx}, V_i \rangle \qquad \text{III-19}$$

where the double subscript implies differentiating with respect to the second subscript.

The three prognostic equations (III-4, III-5 and III-6) are similarly advanced using the Galerkin technique to become, respectively:

$$\langle \dot{\phi}_j V_j, V_i \rangle + \bar{\Phi} \langle D_j V_j, V_i \rangle = - \langle \alpha_j V_{jx}, V_i \rangle - \langle \beta_j V_{jy}, V_i \rangle \qquad \text{III-20}$$

$$\langle \dot{\xi}_j V_j, V_i \rangle = - \langle (uQ)_j V_{jx}, V_i \rangle - \langle (vQ)_j V_{jy}, V_i \rangle \qquad \text{III-21}$$

$$\langle \dot{D}_j V_j, V_i \rangle - \langle \phi_j \nabla^2 V_j, V_i \rangle = \langle (vQ)_j V_{jx}, V_i \rangle - \langle (uQ)_j V_{jy}, V_i \rangle$$
$$+ \langle K_j \nabla^2 V_j, V_i \rangle \qquad \text{III-22}$$

where ∇^2 is the Laplacian operator and the dot implies differentiation with respect to the time dependence in III-4, III-5 and III-6.

Similarly, Galerkin equations are formulated for equations III-7 through III-15.

C. TIME DISCRETIZATION

The equation set III-20, III-21 and III-22 is arranged so that all the terms on the left hand side can be treated implicitly, and all the terms on the right hand side can be treated explicitly. The explicit time integration will be done by the leapfrog difference method. To start the time integration, two forward half steps are taken, after which the full leapfrog scheme is used for the remainder of the forecast period.

The vorticity equation III-21 is solved independently from III-20 or III-22. However, III-20 and III-22 (continuity and divergence equations, respectively) are coupled. To explicitly solve either, decoupling of the equations is necessary. In this thesis this is done through algebraic substitution of III-22 (solved for D(n+1)) into III-20. Once the time integration is performed on III-20, III-22 can be solved for D(n+1) using the ϕ (n+1) value.

The final prediction equations are

$$\begin{aligned}
\phi_j^{n+1}[\langle V_{jx}, V_{ix}\rangle + \langle V_{jy}, V_{iy}\rangle + C\langle V_j, V_i\rangle] = \; & - [BDRY]^{n+1} - [BDRY]^{n-1} \\
& + C\phi_j^{n-1}\langle V_j, V_i\rangle - AD_j^{n-1}\langle V_j, V_i\rangle \\
& - \phi_j^{n-1}[\langle V_{jx}, V_{ix}\rangle + \langle V_{jy}, V_{iy}\rangle] \\
& - 2[(vQ)_j^n\langle V_{jx}, V_i\rangle - (uQ)_j^n\langle V_{jy}, V_i\rangle] \\
& - 2K_j^n[\langle V_{jx}, V_{ix}\rangle + \langle V_{jy}, V_{iy}\rangle] \\
& - 3[(\phi u)_j^n\langle V_{jx}, V_i\rangle + (\phi v)_j^n\langle V_{jy}, V_i\rangle]
\end{aligned} \qquad \text{III-23}$$

where $A = 4/(2\Delta t)$, $B = A/\bar{\Phi}$, $C = B/(2\Delta t)$ and [BDRY] is the geostrophic boundary contribution, see Section G.

$$
\begin{aligned}
S_j^{n+1}\langle V_j, V_i\rangle = {} & S_j^{n-1}\langle V_j, V_i\rangle \\
& - 2\Delta t\left[(uQ)_j^n\langle V_{jx}, V_i\rangle + (vQ)_j^n\langle V_{jy}, V_i\rangle\right]
\end{aligned}
\qquad \text{III-24}
$$

$$
\begin{aligned}
D_j^{n+1}\langle V_j, V_i\rangle = {} & D_j^{n-1}\langle V_j, V_i\rangle + (\Delta t/2)\left[\phi_j^{n+1}\langle \nabla^2 V_j, V_i\rangle\right. \\
& + \phi_j^{n-1}\langle \nabla^2 V_j, V_i\rangle + 2(vQ)_j^n\langle V_{jx}, V_i\rangle \\
& \left. - 2(uQ)_j^n\langle V_{jy}, V_i\rangle + 2K_j^n\langle \nabla^2 V_j, V_i\rangle\right]
\end{aligned}
\qquad \text{III-25}
$$

After these three elliptic equations are solved, the history of the variables III-7 through III-15 is updated.

A large time step can be applied to this form of the shallow water equations due to the semi-implicit nature of the equations. This is very important since finite element methods generally require more computer time per time step. The vorticity-divergence formulation acts as a filter, which slows down the high frequency waves in the solution. The two-dimensional advective stability criterion for a linear element, derived by Cullen (1973), was used to determine the correct time step,

$$
\Delta t = \frac{\Delta x}{|c|\sqrt{6}} \qquad \text{III-26}
$$

where Δt is the time step in seconds, Δx the shortest grid spacing in meters and c the fastest phase velocity.

D. COMPUTATIONAL TECHNIQUES

The final prognostic equation set requires the solution of a Helmholtz equation for ϕ and Poisson equations for Ψ and χ. The most common method of solution used by meteorologists has been the successive over relaxation method (SOR) in which an initial guess of the solution is made and then progressively improved until an acceptable level of accuracy is reached. SOR is employed in the solution of the equations, where III-23 can be represented by

$$\nabla^2[M]\{x\} - C[M]\{x\} = \{b\} \qquad \text{III-27}$$

and III 24, III 25 by

$$\nabla^2[M]\{x\} = \{b\} \qquad \text{III-28}$$

where ∇^2 the Laplacian operator, $[M] = \langle V_j, V_i \rangle$ matrix, $\{x\}$ - the dependent variable in vector notation, C - constant as in III-23 and $\{b\}$ the right hand side of the equation or the forcing function.

The mass matrix [M], dimensioned (nxn), is a matrix of coefficients whose rows are the equations of the system to be solved. There exists a one to one correspondence between the rows of the mass matrix and the nodes of the domain. Each equation has a term (column) for each node, where a non-zero term represents connectivity. Nodes are connected if they are both vertices of the same element. Obviously [M] is a sparse matrix containing the inner products for the

left hand side. Chapter 4 of this thesis will describe the matrix compaction procedure.

The forcing function {b}, dimensioned (nx1), involves only variables at the current time step and is easily computed using four very versatile subroutines described in detail in the next chapter.

The initial guess to start SOR is the previous time step solution. An average of 30 passes per equation are needed for each time step. The solution is considered to have converged to its final value when the residual for each node has been reduced to some acceptably small value.

The diagnostic equations III-7 through III-15 must also be solved every time step. However, the same technique is not used for these equations. Dr. M.J.P. Cullen suggested an under relaxation scheme for which three passes over the domain should produce a solution of acceptable accuracy, since the coefficient matrix is so strongly diagonally dominant. Mass lumping of the coefficient matrix is used for the first guess. This technique requires replacing the mass matrix [M] by the identity matrix [I]. A first guess of this type is able to describe most of the large scale features, which in turn reduces the number of iterative passes over the field. Successive passes converge to solutions which describe smaller scale motion, approximately to the same order of magnitude as introduced by computational error, so that further iterations are not needed.

E. GRID GEOMETRY

The domain of this model is a cylindrical channel, with total length of 4045 Km and width of 3503 Km. The channel simulates a belt around the earth and it proves to be an excellent test bed for comparing with the finite element formulations used by Kelley (1976) and Older (1981).

The domain is subdivided into equilateral triangles as shown in Figure 6. Most of the test runs for this thesis use a 12x12 mesh which has 156 nodes and 288 elements. This implementation is not restricted to one grid pattern. The technique developed by Older (1981) to vary the nodal geometry smoothly to achieve areas of denser and coarser resolution is also implemented, as in a third grid pattern that varies the nodal geometry abruptly. A short discussion of these nodal geometries with accompaning illustrations of each is presented in Chapter VII, where the different test cases are described.

Cyclic continuity is assured in the x direction by wrapping the domain around the earth to form a cylindrical domain. This has the advantage of eliminating the east-west boundaries and it simulates the flow around the earth. The only boundaries on this domain are the north-south walls and their treatment will be discussed shortly.

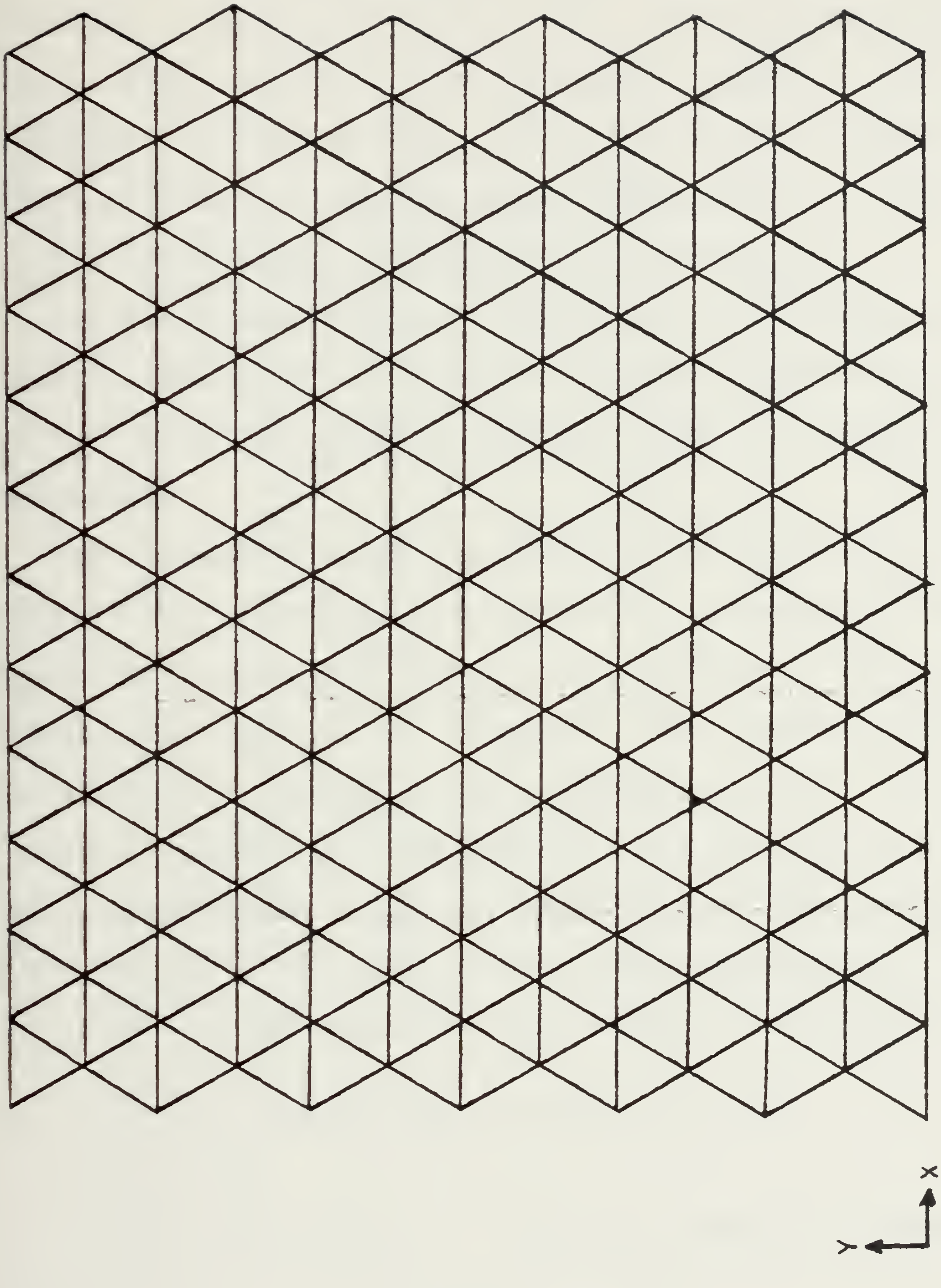


Figure 6 . Domain divided into equilateral triangles.

F. INITIAL CONDITIONS

As mentioned previously, the reformulation of the governing equations into the vorticity-divergence shallow water equation set requires solving the time dependent variables in terms of the stream function and velocity potential. The continuity equation is not altered, so that its solution is expressed in terms of ϕ.

For the basic testing of the model's performance, simple analytic sinusoidal initial conditions are used to insure the most accurate analysis possible and to simplify the comparisons.

The sinusoidal initial fields are graphically shown in Figure 7 as 3-dimensional surfaces. The geopotential field ϕ consist of a half sine wave in the y direction and a single cosine wave in the x direction. The stream function Ψ, calculated by dividing the geopotential field by the coriolis force, has the same physical structure as ϕ. The velocity potential X has a single sine wave in the x direction and a half sine wave in the y direction.

These initial conditions are computed as follows

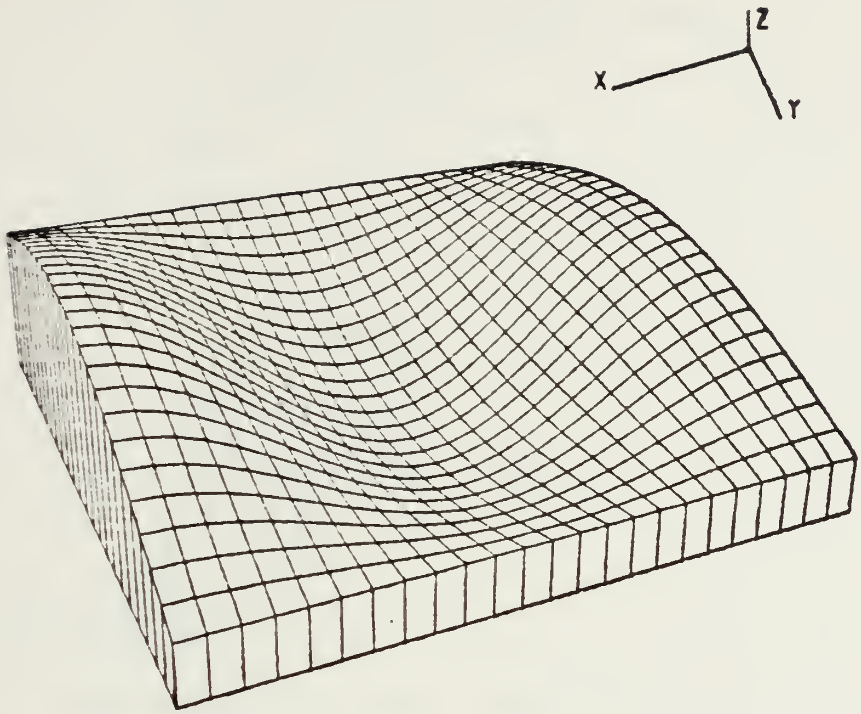
$$\phi = f_0 A \sin\alpha_1 \cos\alpha_2 - f_0 \overline{U}(y - y_m) + \overline{\Phi}$$

$$\Psi = \phi / f_0 \qquad \text{III-29}$$

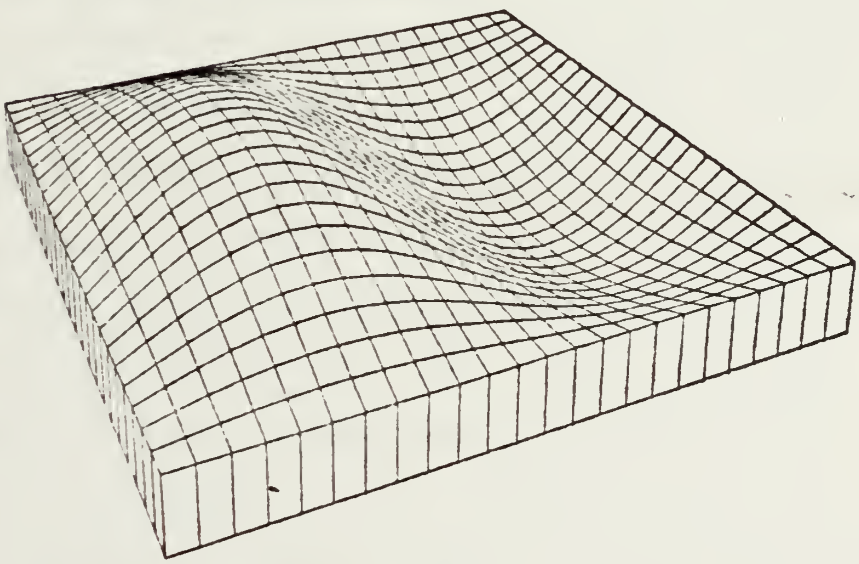
$$X = C \sin\alpha_1 \sin\alpha_2 \qquad \text{quasi-geostrophic divergence}$$

where A - arbitrary amplitude

f_0 - coriolis value for mid-channel latitude



a) ϕ and Ψ initial fields.



b) χ initial field.

Figure . 3-dimensional view of the inital fields.

$\overline{U}$ mean flow

y_m - mid-latitude value of y

$\hat{\Phi}$ - mean free geopotential height

$= 49{,}000\ m^2/s^2$

α_1 - $\pi y/W$

α_2 $2\pi x r/L$

r - wave number

W channel width

L - channel length

C - $-(f_o \overline{U} \alpha_2 BA)/(f_o^2 + \hat{\Phi} B)$

B - $\alpha_1^2 + \alpha_2^2$

G. BOUNDARY CONDITIONS

Boundary conditions are only required on the north and south walls of the grid domain. Due to cyclic continuity, the domain is wrapped around creating a cylinder eliminating the east and west boundaries. However, careful attention to detail is needed during the implementation to assure this continuity. Separate boundary conditions are applied to each of the predictor equations III-23, III-24 and III-25. These conditions are computed for the wall nodes only and are applied during each pass through the relaxation scheme.

The vorticity equation III-24, the most sensitive of the predictor equations to solve, requires Ψ on the north-south boundaries to remain constant for the entire forecast period. Since this equation is solved in terms of Ψ, the initial north-south Ψ values are saved and assigned to the

boundary points after each pass through the relaxation subroutine.

The proper boundary condition for the divergence equation III-25 would be $\partial X/\partial n = 0$. However, for the purpose of this study, there is more interest in the sinusoidal variation in the y direction and not in the region of the walls. Therefore $X = 0$ is appropriate.

The continuity equation III-23, the most complex predictor equation, requires that there be no mass flux through the north-south walls. The geostrophic boundary condition

$$\frac{\partial \phi}{\partial y} = -uf_o \qquad \text{III-30}$$

is applied to the north south boundary nodes for the terms [BDRY] in equation III-23. Integrating the inner product $\langle \phi_j \nabla^2 V_j, V_i \rangle$ by parts produces the boundary terms

$$\iint_{yx} \nabla^2(\phi_j V_j) V_i \, dxdy = \iint_{yx} \nabla \cdot (\nabla \phi_j V_j) V_i \, dxdy$$

$$= \iint_{yx} [\nabla \cdot (V_i \nabla(\phi_j V_j)) - \nabla(\phi_j V_j) \cdot \nabla V_i] \, dxdy$$

$$= \oint V_i \nabla(\phi_j V_j) \cdot \hat{n} \, dr - \iint_{yx} \nabla V_i \cdot \nabla(\phi_j V_j) \, dxdy$$

$$= [BDRY] - \phi_j[\langle V_{jx}, V_{ix} \rangle + \langle V_{jy}, V_{iy} \rangle] \qquad \text{III-31}$$

where $\hat{n}$ is a unit vector normal to the domain and dr is the differential distance along the path of integration on the perimeter of the domain.

The geostrophic boundary condition III-30 is substituted into the contour integral in equation III-31 and put into Galerkin form, in the same way as in the one-dimensional advective equation in Chapter II. The resulting term is derived as follows

$$\oint V_i \nabla(\phi_j V_j)\cdot\hat{n}\, dr = \oint \frac{\partial}{\partial y}(\phi_j V_j)\, V_i\, dx$$

$$= \frac{f_o \Delta x}{3}(u_{j+1} + 2u_j + u_{j-1})_i \qquad \text{III-32}$$

Equation III-32 appears twice in the continuity equation III 23, for time levels (n+1) and (n-1). All values of u are known for time (n-1), since they are saved from the previous calculations. However, u(n+1) has not been computed. To solve for u(n+1), both Ψ(n+1) and χ(n+1) are needed. Ψ(n+1) is solved first from the vorticity equation. χ(n+1) needs ϕ(n+1) as part of its solution and ϕ(n+1) needs u(n+1) in its solution. To avoid this problem, it is assumed that χ(n+1) has a negligible contribution to the solution of u(n+1) and only Ψ(n+1) is used.

IV. COMPUTER IMPLEMENTATION

The formulation and general theory of the finite element method was presented in the previous chapters. The objective in this chapter is to discuss some important computational aspects pertaining to the implementation of the finite element prediction system.

The main advantage that the finite element method has over other prediction techniques is its generality. Conceptually, it seems possible by using many elements, to approximate virtually any surface with complex boundaries and initial conditions to such a degree that an accurate solution can be obtained. In practice, however, obvious engineering limitations arise, a most important one being the cost of the computation. As the number of elements is increased, a larger amount of computer time is required for a forecast. Furthermore, the limitations of the program and the computer may prevent the use of a large number of elements. These limitations may be due to the computer speed and storage availability, or round-off errors propagated in the computations because of finite precision arithmetic. Also, the malfunction of a hardware component, if the prediction is carried out using many computer hours to execute, can be a serious problem. It is therefore desirable to use efficient finite element programs.

The effectiveness of a program depends essentially on the following factors. Firstly, the use of efficient finite element techniques is important. Secondly, efficient programming methods and sophisticated use of the available computer hardware and software are important. The third very important aspect in the development of a finite element program is the use of appropriate numerical techniques.

The vorticity-divergence model described in the previous chapter is implemented on the IBM 3033 computer located at the Naval Postgraduate School. Some notable features of its architecture are the three trillion bytes of virtual mass storage, of which eight mega bytes are available to each user, and the 57 nanosecond machine cycle time. The model is executed mostly using a 12x12 element domain requiring 400k bytes of storage and 30 seconds of CPU time to execute. Exceeding execution time and/or available storage is not a problem, in fact the system allowed a lot of flexibility during the implementation phase of the model.

The source code is written using FORTRAN IV and compiled on an optimizing Fortran H compiler. Appendix A contains the source code listing, which is divided into five subdivisions delineating the logical structure of the program.

A. PROGRAM ARCHITECTURE

Program features incorporated in the model are:

1) Modularity. With only a few exceptions, each module is limited to one page of FORTRAN code. This makes it

easier to comprehend the program. Each module performs only one task. For example, subroutine CONTEQ computes the value of the forcing terms for the continuity equation. Likewise there is also one module for the divergence and vorticity equations. To implement a new set of equations, only these modules would have to be altered.

2) Easily controllable switches. Switches may be set to either print, plot or tabulate harmonic analysis data for most of the available fields. The ability to display intermediate results allows each portion of the algorithm to be monitored for computational adjustments. This also makes it easier for unfamiliar users to become acquinated with the computational model.

3) Forcing term subroutines. In previous implementations, each forcing term was calculated by a special subroutine. In this implementation, the calculations are accomplished by general purpose routines which simplify the implementation of the complex prognostic equations. This allows implementation of different equation sets (i.e. Baroclinic Model) over the same domain with minimal effort.

4) Documentation. Each variable is defined by a short phrase (Appendix A, A.). The function of each module is described in an introductory paragraph. Shared data is placed in named common blocks and identified with each subroutine which uses them. A subroutine index is given.

1. Main Program

The main program is short, calling only six modules which reflect the basic sequential flow of the model. It starts with initialization of all model parameters (i.e. model options, domain, finite element arrays, inner products). It then initializes the input fields (i.e. geopotential heights, stream function and velocity potential) and is followed by initialization of all remaining dependent variables. At this time the model is totally initialized and time integration begins. As mentioned previously two techniques are employed for time integration, each having its own module. Upon completion of the last forecast, the program terminates.

Arrays are the only data structures used and are grouped using 19 different common blocks. Several arrays are used as static link lists, as described in detail later, which simplified the algorithms. The common block format has the advantage of reducing the overall execution time of the program. Most of the arguments passed during a call to a subroutine are contained in common. This requires less time to execute since no parameter passing is required for the arguments. Another benefit of this format is that the code becomes less cumbersome and more readable. Each variable and array is defined in the first subsection of Appendix A along with a page index for the subroutines.

2. Initialization Phase

Appendix A, Section C contains all the subroutines used during the initialization phase of the program. From the user point of view, the most important subroutine is INITG3, the first subroutine called, which contains all the global variables that control the different options available per run. This is the only subroutine that is changed to run the different experiments, assuming that no new computational technique is introduced. The selection of options are:

1) - channel location - the channel may be shifted north or south by presetting the north/south latitude limits in INITG3.

2) - variable geometry - the node positions may be grouped for more dense node patterns to yield higher resolution. Two variables R1 and R2 set the ratio used to vary the spacing along the x and y axes, respectively.

3) - initial field wave length and amplitude can be altered to produce various effects.

4) change the initial mean flow.

5) - diffusion can be entered for any of the three prognostic equations.

6) - maximum length of forecast period may be changed and a print, plot or harmonic

analysis of any dependent variable may be requested for any time interval.

Once the experiment is determined, the options listed above are set. The program is ready to be executed.

The largest part of the initialization phase consists of establishing the domain and producing all the finite element computational vectors that remain constant throughout the experiment.

The first several steps in setting up the domain are concerned with indexing. Subroutine CORRES is called first, where all the nodes (grid points) and elements (triangular areas) are numbered consecutively starting at the southwest corner of the domain and moving eastward across each row or latitude. For each element, a record of all of its nodes (vertices) are stored in array ELMENT (M,3), where M is the total number of elements. To facilitate the inner product evaluation later, a local numbering system is required for each element. That is, for each element, its nodes are stored counterclockwise in a positive sense. The first node however, is arbitrary.

With the domain divided and numbered, a connectivity list (the correspondence between each node and the neighbor nodes) is constructed for each node by subroutine CORREL. Each node is adjacent to four or six other nodes depending on whether it is a boundary or interior node, respectively. These adjacent nodes, plus itself, make up the connectivity

list for one node. The connectivity lists are then concatenated sequentially starting with the first nodes connectivity list into the vector NAME (NN), where NN is the sum of the nodes in each connectivity list. (i.e. for a 12x12 domain with 156 nodes, and equilateral elements NN = 1044). For the first time during the initialization phase, special attention is given to cyclic continuity. As discussed earlier, cyclic continuity is the joining of the east and west boundaries to create a cylindrical channel. The connectivity list for these east/west boundary nodes must be complete to insure proper continuity for the calculations later.

The connectivity vector NAME is frequently used during most computations. Two utility vectors ISTART (containing the starting location in NAME for a particular node) and NUM (containing the number of nodes in its connectivity list) are used to locate and index through the vector NAME, as will be seen shortly. This same technique is used to index through the coefficient matrices and used during most of the node interaction computations.

The physical properties of the channel are calculated next in subroutine CHANAL. Here the north and south latitude boundaries, which were pre-set in INITGB by the user, are used to compute the grid spacing along the x and y axis. Since this channel simulates a belt around the earth, the magnitudes of both DELTAX and DELTAY (meters) are

proportional to the width of the channel divided by the number of grid points in the y axis.

The Cartesian coordinates for each node are computed by subroutine LOCATE using the DELTAX and DELTAY calculated in CHANAL. If the option to use varying grid geometry is desired, subroutine TRANS transforms the grid geometry. TRANS also computes the corresponding new Cartesian coordinate values for each grid point and calculates the minimum DELTAX and DELTAY within the domain. When the geometry is changed to create a smaller DELTAX or DELTAY, the two dimensional advective stability criteria is also changed. A new time step DT has to be computed using equation III-26. Since TRANS transforms the geometry, it also computes the new DT.

Another transformation is required as discussed in Chapter II. The transformation from Cartesian coordinates to area coordinates is needed to perform the area integration of the inner products. Subroutine AREA computes these transformations exactly as outlined in Chapter II, Section C. Again cyclic continuity is very important and special care is needed to insure proper transformation.

Following the area transformation is the computation of all the inner products that are required to solve the equations. The advantage of using area coordinates is that the inner products (function of space coordinates only) are computed and stored once and used repeatedly without

recalculation. Subroutine INNER computes and stores these products using the formulas derived in Chapter II.

The coefficient matrix, dimensioned NxN, where N is the total number of nodes, is a matrix of coefficients whose rows are the equations of the system to be solved. As discussed in the computational technique section, the members of this sparse matrix are the inner products for the left hand sides of the equations. Three coefficient matrices are used in the solution of the equations. The diagnostic equations (III-7 through III-15) use a coefficient matrix with the inner product $\langle V_j, V_i \rangle$ which is constructed by subroutine AMTRX1 and stored in compacted form in vector G(NN) by subroutine ASEMBL. However, when solving the prognostic equations, these coefficient matrices have a DT (time step in seconds) term, so that these matrices are not assembled until the time integration begins. The vorticity and divergence equations (III-24, III-25) use the coefficient matrix E(NN) with inner products $\langle V_{jx}, V_{ix} \rangle + \langle V_{jy}, V_{iy} \rangle$ in solving the Poisson equations for the stream function and velocity potential, respectively. The continuity equation (III-23) uses a combination of inner products in its coefficient matrix F(NN) as follows

$$\langle V_{jx}, V_{ix} \rangle + \langle V_{jy}, V_{iy} \rangle + \frac{4}{\Phi(\Delta t)^2} \langle V_j, V_i \rangle \qquad \text{IV-1}$$

to solve the Helmholtz equation. At the start of each time integration module, subroutine AMTRX2 is called to construct the two coefficient matrices E and F.

These banded and sparse matrices are compacted into vectors to save storage during their assemblage by ASEMBL. The vectors are dimensioned NN, as is NAME, the connectivity vector, and both use ISTART and NUM to index through them. This compaction routine was used by Kelley (1976) and Older (1981) in their models, but was developed by Hinsman (1975).

To illustrate matrix assemblage using an element by element technique, consider Figure 8. Note that this illustration is for element number 3, but all elements are treated in a similar maner. The computational technique required that for each point (node) describing element 3, namely nodes 2, 3 and 14 stored in array ELMENT, the inner product $<V_j, V_i>$ between those points be distributed to their proper location in the coefficient matrix.

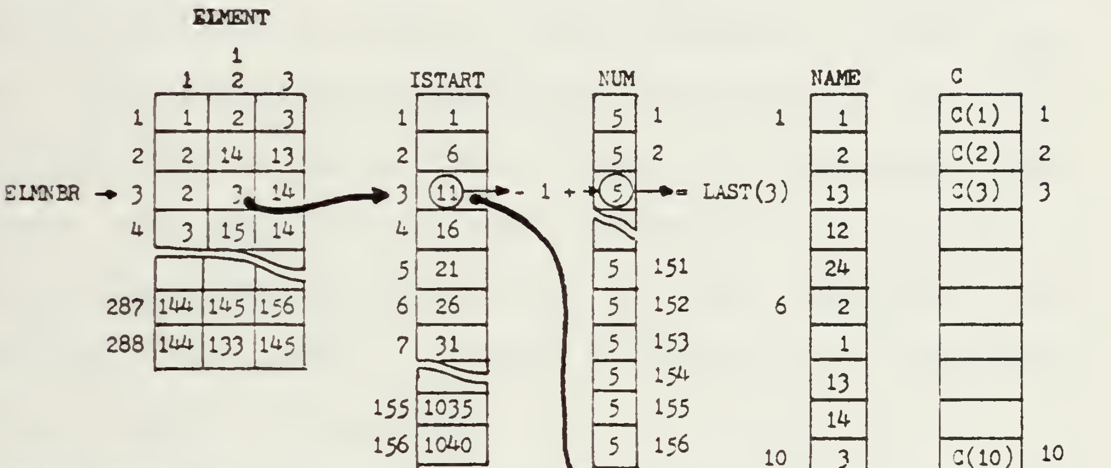
Subroutine AMTRX1 builds the inner product nodal interaction and stores it in matrix B, dimensioned 3x3. Figure 8 illustrates the B matrix for element 3, where the inner product $<V_j, V_i>$ is the multiplicand of the corresponding basis and test functions, respectively.

The local dispensing of interactions is done in ASEMBL. Consider the second row of [B] in Figure 8. These are the interactions between node 3 of element 3 to the test

AMTRX - builds [B] for one element, passes [B] and the element number associated with [B] to ASEMBL. This process continues till all element node interactions are assembled in the coefficient matrix.

$$[B]_3 = \begin{bmatrix} V_2V_2 & V_2V_3 & V_2V_{14} \\ V_3V_2 & V_3V_3 & V_3V_{14} \\ V_{14}V_2 & V_{14}V_3 & V_{14}V_{14} \end{bmatrix} \qquad \text{ELMNBR} = 3$$

ASEMBL - assembles the node interactions into the coefficient matrix [C] , which has the same structure as NAME. The following diagram assembles inner product V_3V_{14} into the coefficient matrix [C], for element 3.



ASEMBL pseudo-code

```
do  i  = 1 → 3                 { 2
    ii = ELMENT(i)             { 3
                               { 14
    do  j  = START(ii) → LAST(ii)
        jj = NAME(j)
        do  k  = 1 → 3         { 2
            kk = ELMENT(k)     { 3
                               { 14
            if ( kk = jj )  then
               C(j) = C(j) + B(i,k)
            else  continue
        end  do
    end  do
end  do
```

Figure 8. Assembling and storing the coefficient matrix for element 3.

functions. ASEMBL locates nodes 3's connectivity list in NAME using ISTART and NUM. In Figure 8, this list is delineated by START(3) and LAST(3). Now ASEMBL steps through the connectivity list for three iterations. During each pass. ASEMBL is searching for one of the three node numbers for element 3. When a match is found with one of element 3's nodes (i.e. 2,3 or 14) and node 3's connectivity list (i.e. 3,2,14,15 or 4) the proper position, to which this interaction is to be added has been found in the coefficient matrix. Since NAME and vector C, the compacted coefficient matrix, are dimensioned identically, the same pointer (i.e. j in Figure 8) is used to index through both arrays. This procedure is repeated for all elements in the domain to assemble the coefficient matrix of the equations. The pseudo-code for ASEMBL is shown in Figure 8 to facilitate stepping through this example.

The domain and all finite element work vectors are initialized at this point. Subroutine ERMSET is called later to compute interpolation points for the harmonic analysis subroutines.

The last phase of the initialization process is the initialization of the dependent variables. The three input fields geopotential heights, stream function and velocity potential are computed in subroutine IC using the equation set III-30. However, the variables calculated from the diagnostic equations have to be computed using the input

fields. These variables are used during each time step while solving the prognostic equations.

The diagnostic equations are solved in subroutine DEPVAR, first during the initialization phase and later during the time integration phase. Each diagnostic equation calls its own module to compute the value of the forcing function and stores the computed values in the vector RES. These equations all use the same coefficient matrix when solving the diagnostic equations. Subroutine SOLVER is sufficiently genereal to solve each equation. SOLVER uses vector RES and coefficient matrix G to under-relax towards the solution. As mentioned previously, the coefficient matrix is strongly diagonally dominant so that three passes over the domain are sufficient. At the end of DEPVAR, output is generated depending on what print, plot, or harmonic analysis controls were preset.

This completes the initialization phase of the model and the program for the forecast phase will be described next.

3. <u>Forecast Phase</u>

The forecast phase is accomplished in two steps. The first time step is made using two half steps by subroutine MATZNO. Here the prognostic coefficient matrices are constructed using half the DT value by calling AMTRX2. AMTRX2 uses the same computational technique to construct the coefficient matrices as described for AMTRX1.

Each of the prognostic equations III-23, III-24 and III-25 calls its own subroutine (CONTEQ, VORTEQ and DIVEQ respectively) to compute all the terms on the right hand side, which are stored in the vector RES. After computations for RES are completed, subroutine RELAX solves the equations by over-relaxation as described in the computational technique in Chapter 3. Once the solutions for the (n+1) time step are completed, DEPAR is called to update the variables from the diagnostic equations. Two passes through MATZNO advances the solution fields one time step.

The remainder of the forecast period is integrated using the leapfrog scheme. Subroutine LEAPFR performs this integration using the identical format as MATZNO, except that DT equals two DT. At preset times as specified in INITGE, the different fields are saved for printing. This process continues until the final forecast time is reached.

B. UTILITY MODULES

Once the equation formulation is completed, as in Chapter III, all the inner products and types of integrations are known. Versatile modules can be written to perform these computations. Consider a term of general form $\langle A_j V_j, V_i \rangle$ where i is the node about which the term is evaluated and the j's are the nodes connected to node i, or the surrounding nodes. The inner product values $\langle V_j, V_i \rangle$ are already computed and stored for all the nodes, during the initialization phase of the model.

The remaining computation to complete the evaluation of this term is the multiplication of the scalar coefficient of A at node j with the corresponding inner product $\langle V_j, V_i \rangle$ for node j. This requires indexing through node i's connectivity list stored in vector NAME, and for each node in the list multiply and total the products. The cumulative sum of these multiplications is assigned as node i's contribution for this term. Subroutine TERM3 performs this exact computation. All that is passed to TERM3 is the scalar field A and the sign of the inner product, TERM3 then computes the contribution for each node in the domain and accumulates it in the work vector RES.

Three other utility modules are; TERM1, which computes the first scalar multiplication for triple inner products (i.e. $\langle A_j V_j B_k V_k, V_i \rangle$). The product $\langle V_j V_k, V_i \rangle$, is again already computed and stored by subroutine INNER. TERM1 computes $\langle B_k V_j V_k, V_i \rangle$ and constructs a compacted vector similar to the coefficient matrices. This reduces the effort of multiplying the second scalar to a TERM3 computation. TERM2 computes node interaction of the following type $\langle A_j V_{jx}, V_{ix} \rangle$, where both the basis and the test functions are derivatives. Lastly, subroutine TERM4 computes node interaction for terms as $\langle A_j V_{jx}, V_i \rangle$, where only the basis function is a derivative.

When examining the right hand side of the equation sets III-23, III-24 and III-25, it is obvious this implementation is a subscripting nightmare; however, the use of the utility

modules TERM1, TERM2, TERM3 and TERM4 simplifies the implementation to only determining what order to call the utility modules. Examination of subroutines CONTEQ, VORTEQ and DIVEQ, which compute the right hand sides for III-23, III-24 and III-25 respectively, illustrates this fact. No other subroutines or calculations were required. Implementation of these equations required minimal effort.

V. PRIMITIVE MODEL EXPERIMENT

The previous two chapters presented the detailed formulation and implementation of the vorticity-divergence, shallow water equation model. The results from this model will be compared with the results from the primitive model in Chapter VII. To facilitate interpretation of the comparisons, a brief description of the primitive model follows. See Kelley (1976) for a detailed discussion of the entire model.

Also presented in this chapter is an experiment which demonstrates significant improvement of the solution from the primitive model. Kelley's implementation used elements which were right triangles. Older (1981) showed that equilateral elements are far superior to triangular elements. This experiment re implements the primitive model using equilateral elements and a comparison is made between the results of both implementations.

A. MODEL DESCRIPTION

A form of the barotropic, shallow water, primitive equations developed by Phillips (1959) is used as the governing equation set for this model. In Cartesian coordinates the equation set is

$$\frac{\partial \phi}{\partial t} = - \frac{\partial}{\partial x} (u\phi) - \frac{\partial}{\partial y} (v\phi) \qquad \text{V-1}$$

$$\frac{\partial u}{\partial t} = - \frac{\partial \phi}{\partial x} - u \frac{\partial u}{\partial x} - v \frac{\partial u}{\partial y} + vf_o \qquad \text{V-2}$$

$$\frac{\partial v}{\partial t} = - \frac{\partial \phi}{\partial y} - u \frac{\partial v}{\partial x} - v \frac{\partial v}{\partial y} - uf_o \qquad \text{V-3}$$

The finite element formulation of this set of equations evaluates the height and velocity components at the same nodal points. This is an important consideration, because the other models (the linearized model, see Chapter VI, and the vorticity-divergence model, see Chapter III) either stagger the dependent variables or have the property of a staggered formulation. When comparisons are made between the models, it is this lattice structure that is being compared.

This form of the shallow water equations includes gravity waves as a solution. Gravity waves have a maximum phase speed of about 300 meters/second. When the correct time step is calculated using equation III-26, a considerably smaller time step is obtained compared to the larger time step permitted in the vorticity-divergence formulation. This is an important feature. If solutions from all models are equally as good, the best formulation would be determined using the computational time required to produce the desired forecast.

All models use the same domain structure. In fact, the domain described in Chapter III was patterned after the domain implemented by Kelley. Again, this domain simulates a belt around the earth, with cyclic continuity which eliminates the east-west boundaries. Rigid boundary walls exist at the equator and at 30 degrees north latitude. The domain is composed of a 12x12 point mesh and subdivided into the right triangular elements illustrated in Figure 9. Notice that the grid points are not shifted as in the equilateral element implementation shown in Figure 6.

The following boundary conditions are imposed:

1) - no cross channel flow at the latitude boundaries.

2) - a geostrophic balance at the channel walls imposed on the continuity equation V-1.

This model has a simple second order diffusive term in the equations of motion V-2 and V-3. However, for the purpose of evaluating these different formulations, this option was not implemented during the comparison phase.

Initial conditions consist of a single wave in the x direction and a half wave in the y direction. The initial fields for the three dependent variables are shown in Figure 12. The maximum zonal wind perturbation of 5.5 meters/second is superimposed on a mean zonal flow of 10 meters/second.

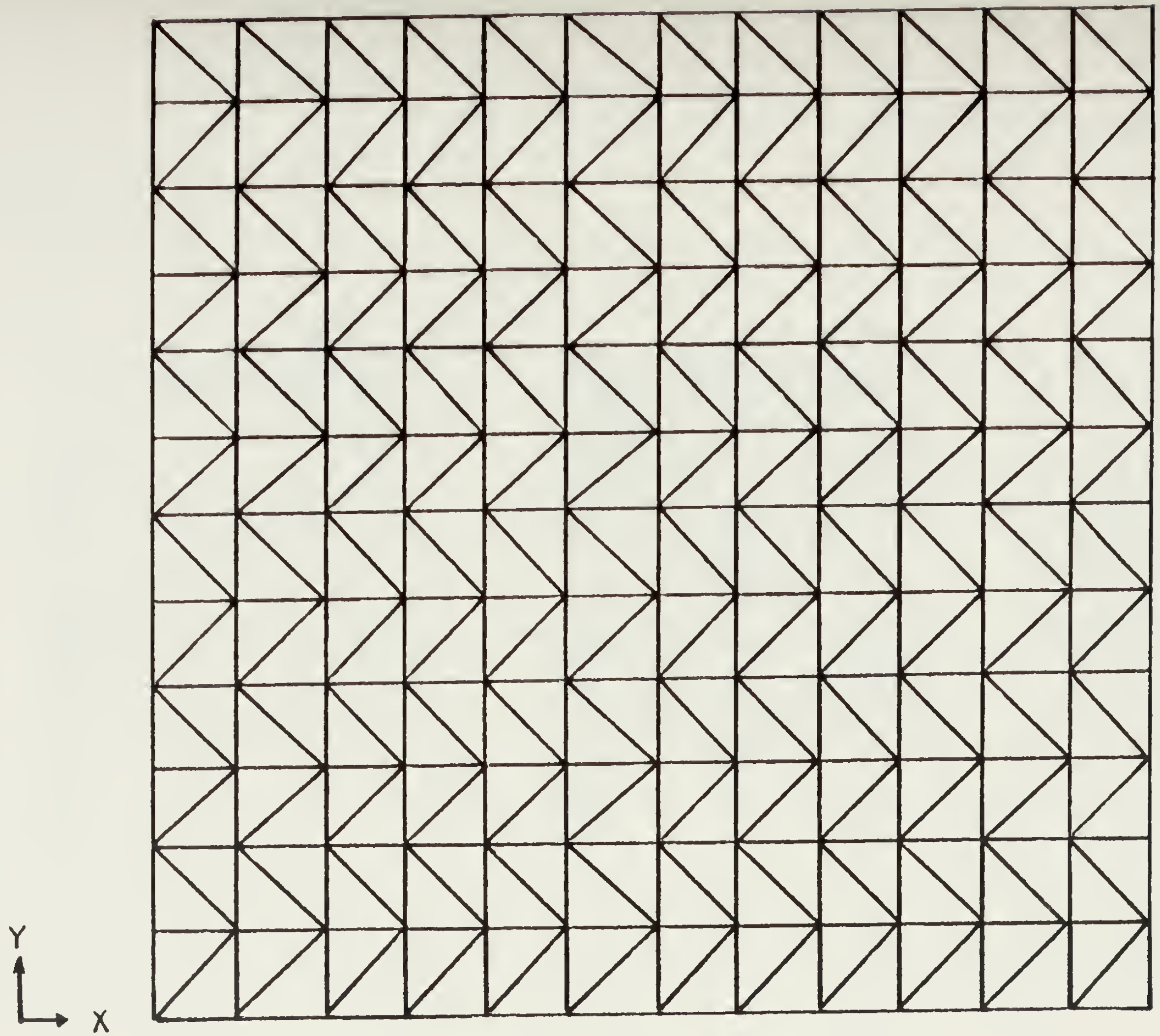


Figure 9 . Domain divided into right triangles.

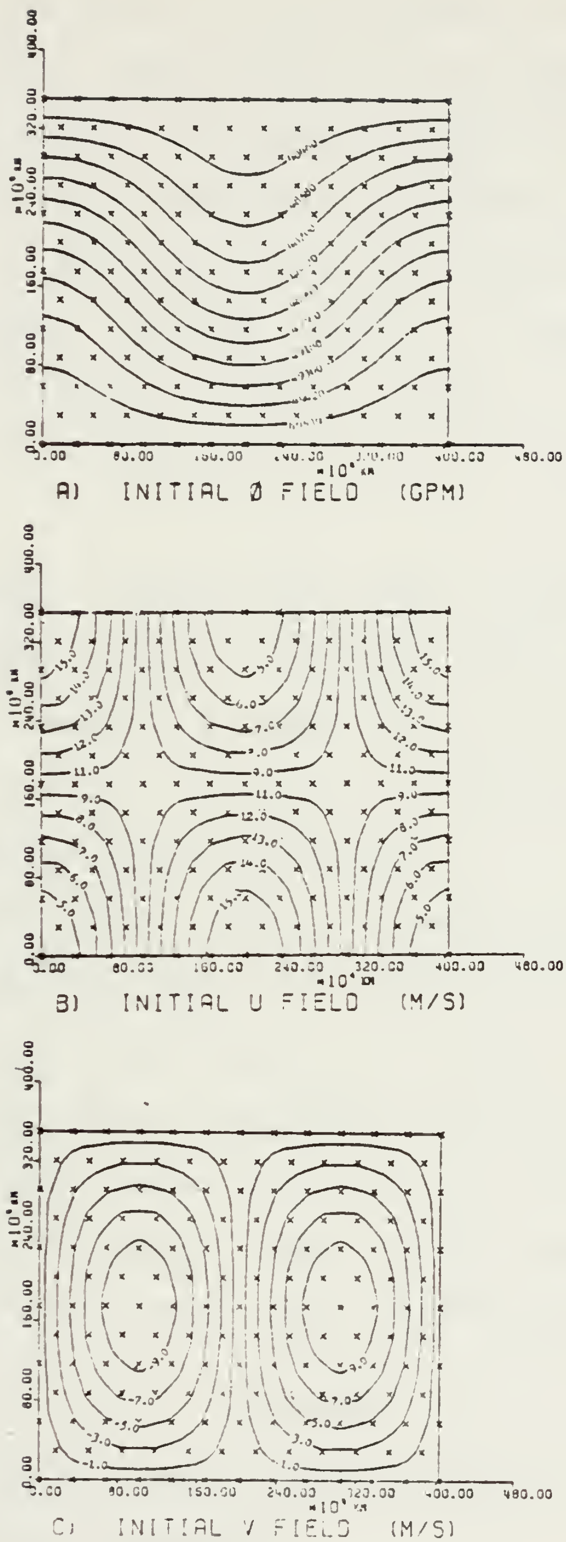


Figure 10. Initial fields for the primitive model. Both the x and y axes are multiplied by 10^4 Km.

This cursory description of the primitive model mentions only those significant features that will weigh heavily in the comparisions later. The Galerkin implementation of this model is similar to that presented in Chapters II and III and the system of equations is solved using a Gauss-Seidel iterative procedure. Further details concerning this primitive model are given in Kelley (1976).

B. RESULTS

This experiment involves the shape of the elements. As mentioned previously, Kelley's implementation subdivides the domain into right triangular elements, as illustrated in Figure 9. Considerable small-scale noise was observed by Kelley in the 48 hour forecast solution.

The transition from right triangles to equilateral triangles changes the size of the domain. With right triangles, the Δx and Δy grid spacings are equal (300 KM). A 12x12 grid matrix has a length and width of 3600 KM. With equilateral triangles, the Δx and Δy grid spacings are no longer equal. Arbitrarily, the Δy grid spacing is held constant (300 KM) and a new Δx grid spacing computed by

$$\Delta x = \Delta y / \cos(30) \qquad \text{V-4}$$

A 12x12 grid matrix with equilateral elements has a width of 3600 KM and a length of 4045 KM.

Figure 11 contains the 48 hour forecasts produced using both types of elements. The three dependent variables fields

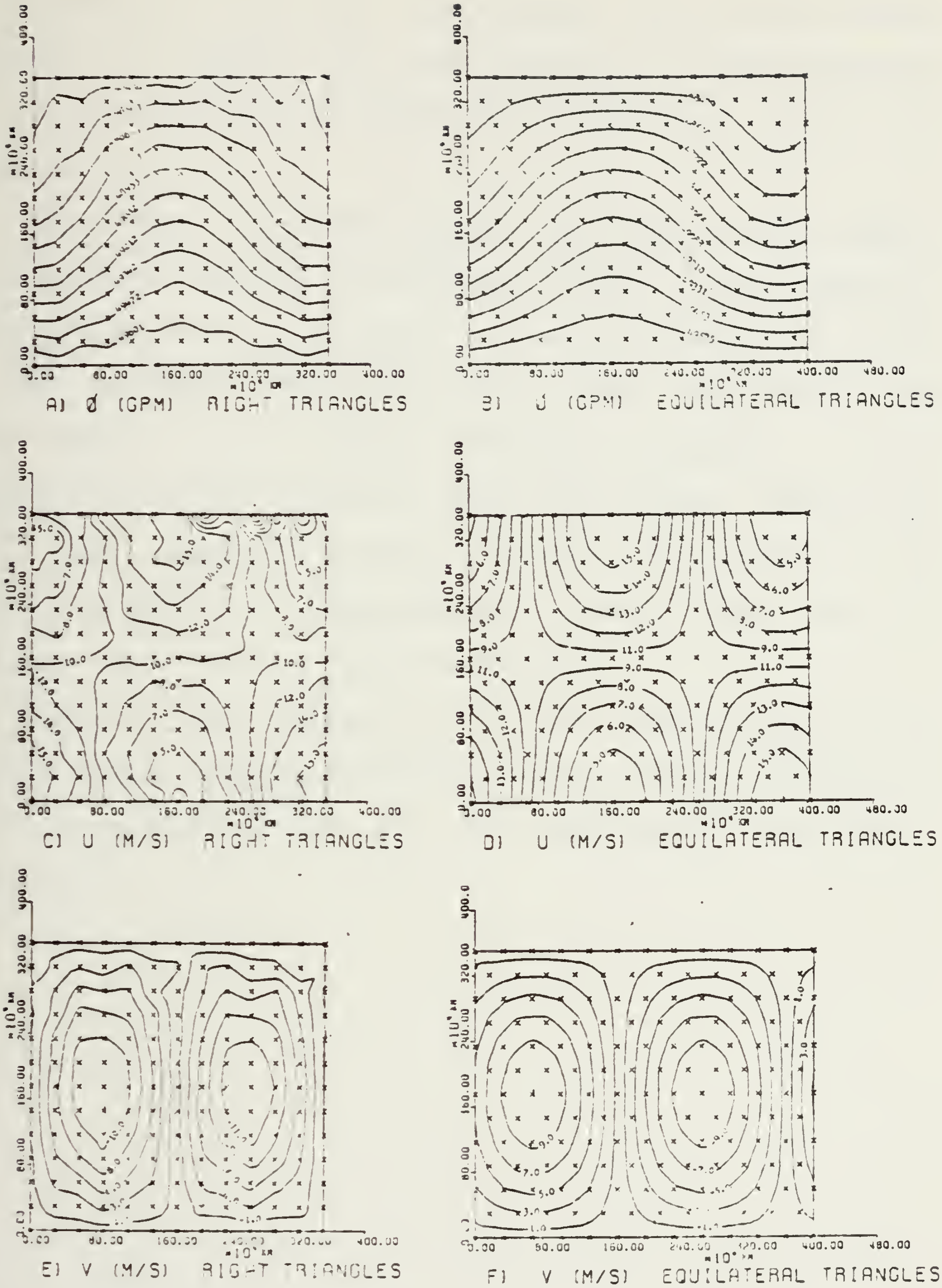


Figure 11. 48 hour forecast comparison from the primitive model using both right triangles and equilateral triangles for element shapes. Arbitrary perturbation amplitude of 5.5 m/s.

are compared for each. The small-scale noise that Kelley observed is present. The three fields show varying degrees of distortions, which are especially noticeable along the boundaries.

Older (1981) found that the root mean square error (RMSE) was reduced 20 percent by using equilateral shaped elements. This improvement is apparent on viewing Figures 11b, d and f. The contours are smooth and the boundaries are noise free. Kelley showed excellent treatment of wave propagation by this primitive model. The lowest resolution grid (6x6) tested by Kelley was within four percent of the true phase velocity. Changing the element shape had no apparent effect on the phase velocities.

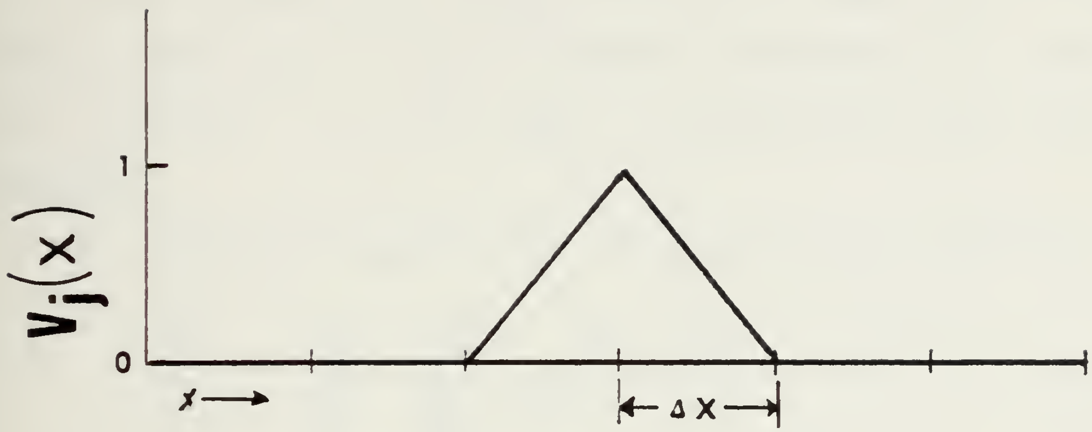
Because the outcome of this experiment was a significantly improved forecast solution, future comparisons with the primitive model will be made using equilateral elements.

VI. LINEARIZED MODEL EXPERIMENT

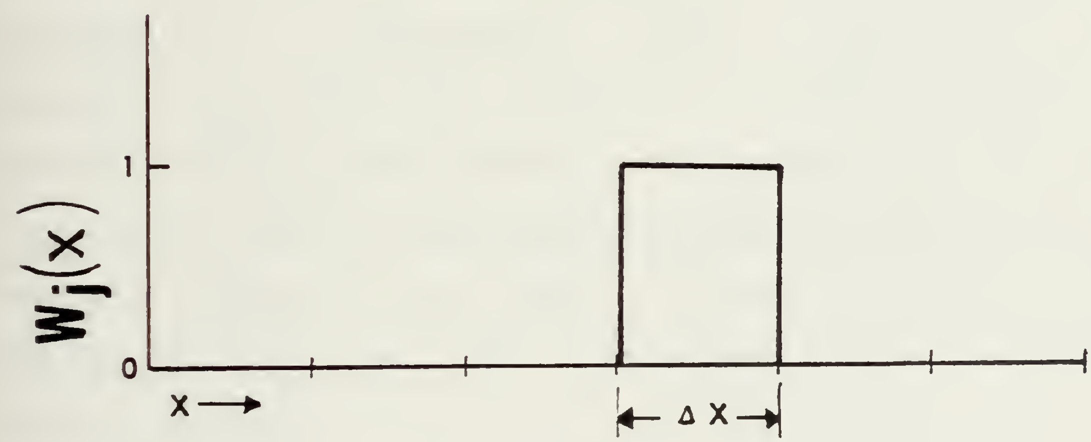
The previous chapter demonstrated how the shape of the element can significantly improve the solution. Williams and Zienkiewicz (1981) used differently shaped basis functions on a linearized equation set to produce excellent solutions when applied to the geostrophic adjustment problem.

Spatial staggering of dependent variables in finite difference formulations has given much better solutions to the geostrophic adjustment process, and these forms are widely used in meteorology. Schoenstadt (1980) found similar results with finite element formulations with piecewise linear basis functions. However, staggering nodal points is not a convenient method to implement, especially in two-dimensions with irregular boundaries, so alternative schemes are needed.

The implementation of the alternative scheme introduced by Williams and Zienkiewicz (1981) are presented in this chapter. As mentioned above, this formulation uses different basis functions for the height and the velocity fields. One of the basis functions is piecewise linear, while the other is piecewise constant, as is illustrated in Figure 12 for a one dimensional domain.



a) Piecewise linear basis function.



b) Piecewise constant basis function.

Figure 12. Different shaped basis functions.

A. EQUATION REFORMULATION

The primitive form of the shallow water equations presented in Chapter V (V-1, V-2 and V-3) is used to derive the linearized equations needed for this experiment. The velocity equations V-2 and V-3 remain unchanged and a linear basis function (V_j) is used to approximate the u and v variables.

The continuity equation V-1 is linearized as follows:

$$\frac{\partial \phi}{\partial t} = \underbrace{- \cancel{u\frac{\partial \phi}{\partial x}} - \cancel{v\frac{\partial \phi}{\partial y}}}_{\text{negligible}} - \bar{\Phi}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \qquad \text{VI-1}$$

where $\bar{\Phi}$ is the average geopotential over the domain. A piecewise constant basis function (W_j) is used to approximate the geopotential. This linearization is reasonable in this case because the Rossby radius of deformation $\bar{\Phi}^{1/2}/f_0$ is much larger than Δx [see Williams and Zienkiewicz (1981)]. The Galerkin method is applied to this linearized equation set using a piecewise linear test function for V-2 and V-3, and a piecewise constant test function for VI-1.

The piecewise constant basis function has the property of displacing the geopotential to the centroid location of the elements, which should give the same effect as staggering the grid points, as seen in Figure 13. The density of geopotential data in the domain is now greater

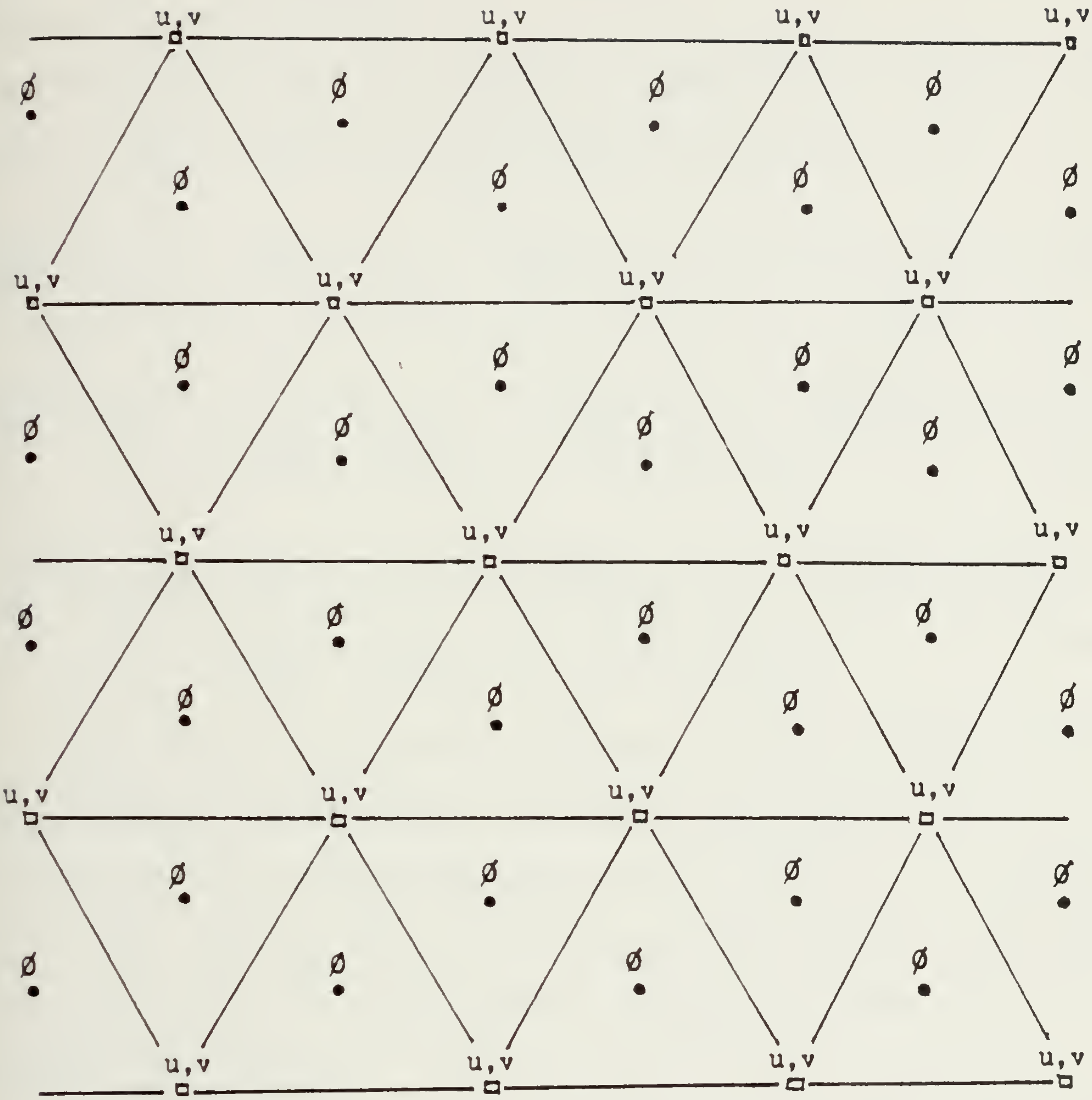


Figure 13. Staggering of the geopotential about the velocity. The ▫ symbol are the actual grid point locations and the • symbol the centroid location of each element.

than the density of the velocity data. Instead of a geopotential value for each node, there is one averaged value over each element.

The final form of the Galerkin equation for VI-1, V-2 and V-3 after performing the time differencing is:

$$\phi_j^{n+1}\langle W_j, W_i\rangle = \phi^{n-1}\langle W_j, W_i\rangle + \Delta t\bar{\Phi}[u^n\langle V_{jx}, W_i\rangle + v^n\langle V_{jy}, W_i\rangle] \quad \text{VI-2}$$

$$u_j^{n+1}\langle V_j, V_i\rangle = u^{n-1}\langle V_j, V_i\rangle - \Delta t[\phi_j^n\langle W_{jx}, V_i\rangle + u_j^n u_k^n\langle V_j, V_{kx}, V_i\rangle$$
$$+ v_j^n u_k^n\langle V_j, V_{ky}, V_i\rangle - f_o v_j^n\langle V_j, V_i\rangle] \quad \text{VI-3}$$

$$v_j^{n+1}\langle V_j, V_i\rangle = v_j^{n-1}\langle V_j, V_i\rangle - \Delta t[\phi_j^n\langle W_{jy}, V_i\rangle + u_j^n v_k^n\langle V_j, V_{kx}, V_i\rangle$$
$$+ v_j^n v_k^n\langle V_j, V_{ky}, V_i\rangle - f_o u_j^n\langle V_j, V_i\rangle] \quad \text{VI-4}$$

This linearized set of equations VI-2, VI-3 and VI-4 is solved using a Gauss Seidel iterative procedure. It is worth mentioning that the coefficient matrix $\langle W_j, W_i\rangle$ in equation VI-2 has all non zero coefficients equal to one, since the integration of the inner product $\langle W_j, W_i\rangle$ involves piecewise constant functions.

This equation set is implemented rather than the equation set V-1, V-2 and V-3 using all of the existing primitive model code. The major modification involved the way that the geopotential was referenced. With an average geopotential over each element instead of a value at each node for a 12x12 mesh, there are 288 geopotential points versus the 156 velocity (u,v) points.

B. RESULTS

The results from this linearized model are compared to those from the primitive model. The initial field for this experiment, Figure 14 has a maximium perturbation zonal wind that is one fifth of the value used in Chapter V, Figure 10. The mean zonal wind remains 10 meters/second. The 48 hour forecast solutions for each field are compared in Figure 15.

This alternative formulation shows some promise, although there are some minor perturbations in these contours compared to those in the primitive solution. No explanation is offered for this small-scale noise, although possibly the other formulations presented by Williams and Zienkiewicz (1981) would improve the solutions.

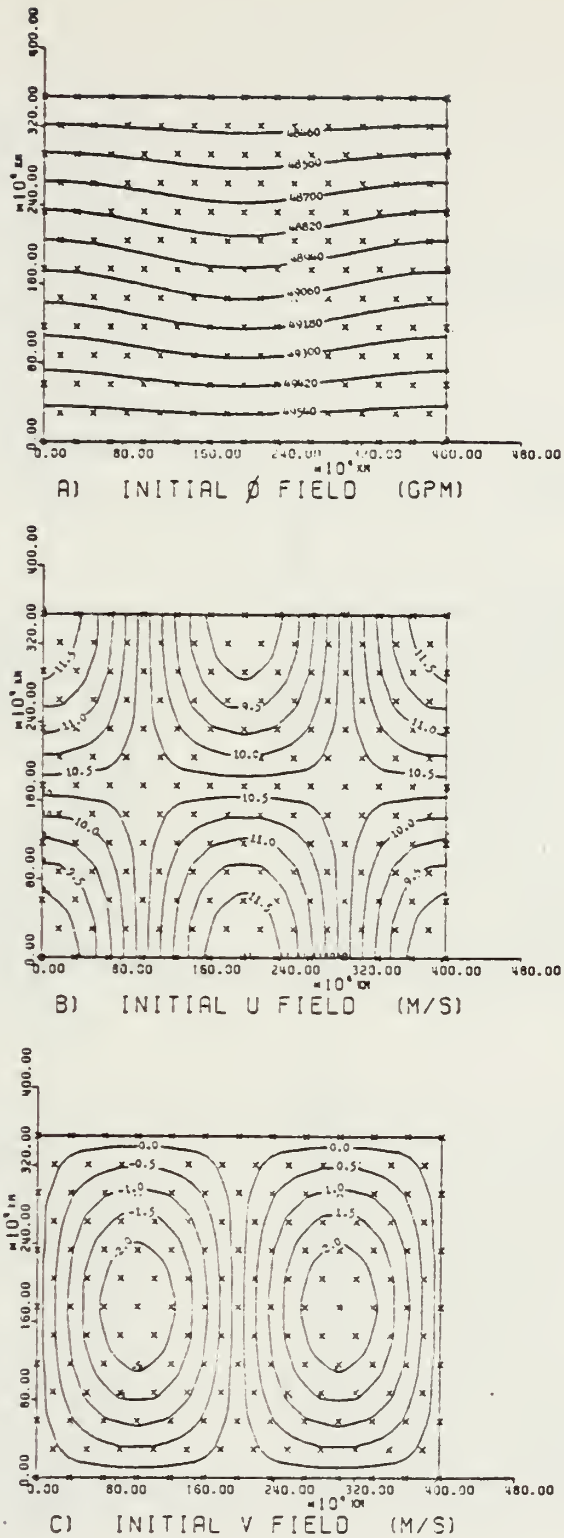


Figure 14. Initial fields for both the primitive model and the linearized model.

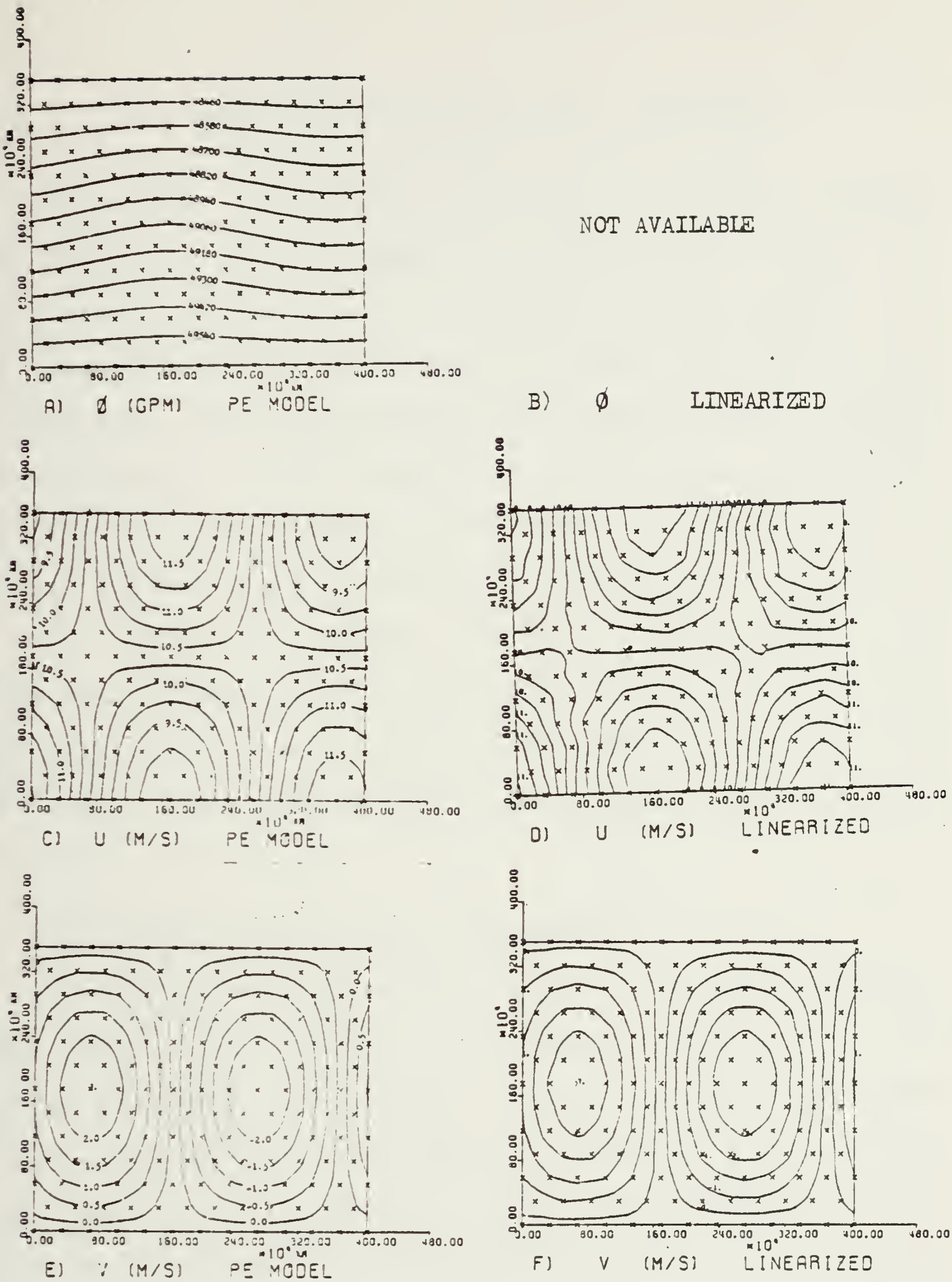


Figure 15. 48 hour comparison between the primitive model and the Linearized model. (APA = 1.1 m/s)

VII. VORTICITY-DIVERGENCE MODEL EXPERIMENT

As in the previous two chapters, a comparison between two models will be presented. The results from the vorticity-divergence model are compared to the results from the primitive model. To fully exploit the differences between the models performances and indicate the strengths and weaknesses of each formulation, three domain geometries and three initial conditions are used. All solutions are at 48 hour, except for one case which was extended to 96 hours.

From these two finite element formulations some additional insight is obtained concerning the execution time required as the grid resolution changes. Lastly, a brief discussion on the sensitivity of the computational technique is given.

A. TEST DOMAINS AND INITIAL CONDITIONS

The three domain geometries used in the model evaluation are illustrated in Figure 16. All domains consist of a 12x12 element mesh with equilateral shaped elements (156 grid points) and cyclic continuity is imposed on the east and west boundaries. The domain has dimensions of 4045 KM along the x axis and 3503 KM along the y axis.

The regular domain (Figure 16a) has a uniform distribution of grid points, with a minimum grid spacing

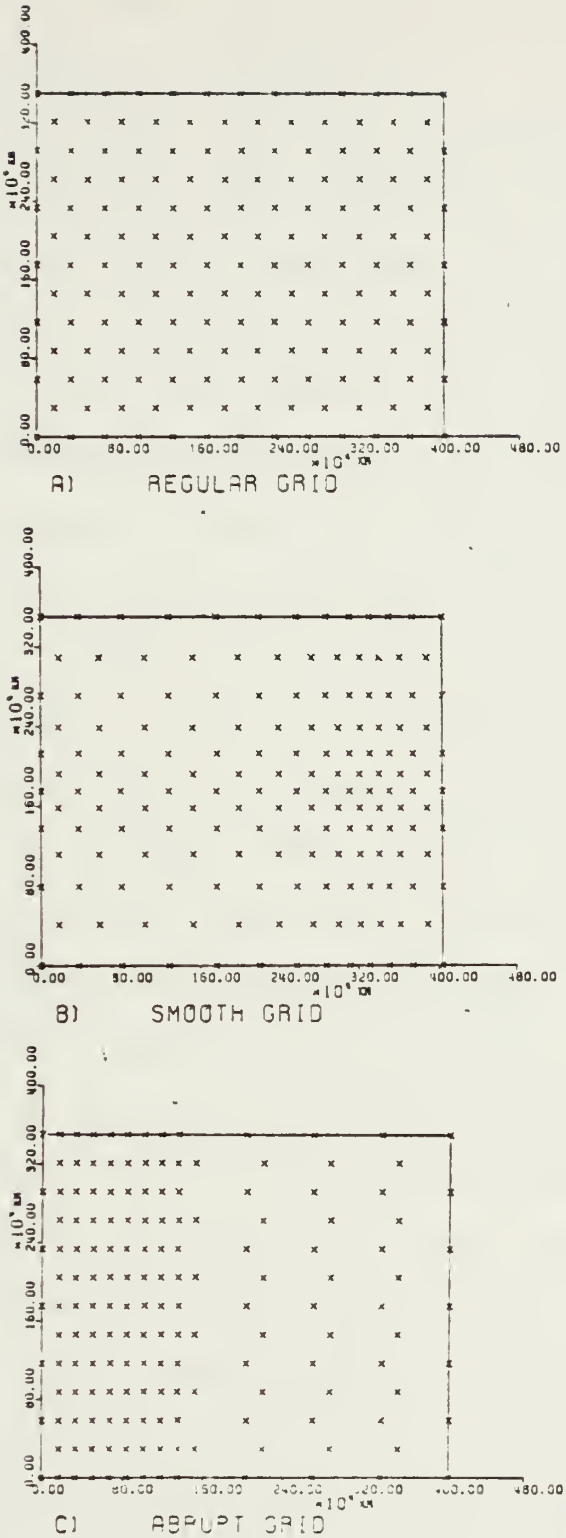


Figure 16. Test domain geometries.

along the x axis of 337 KM. This is the most congenial domain and produces the best results.

The smooth domain (Figure 16b) has a smoothly varying distribution of grid points. This technique allows a smooth variation of resolution. It was developed by Older (1981), who showed how it significantly reduced noise at all frequencies compared with other variable grid domains.

The degree of resolution variation is accomplished by selecting an appropriate value for the ratio of maximum stretch to minimum shrink along both axes. The experiments presented in this chapter use a 2.5 ratio for both directions. This produces a grid point concentration in the right center of Figure 16b and coarse resolution at the top and bottom left of the domain. The minimum grid spacing along the x axis is 199 KM in the dense grid point area.

The third domain was the least hospitable geometry for both models. The abrupt domain (Figure 16c) has a dense grid point concentration on the left and coarse spacing on the right of the domain. The minimum grid spacing along the x axis is 168 KM in the area on the left. The grid spacings are uniform except for the abrupt change along the center.

Although these three domains are simple in structure, they are adequate to test both models' performance. To further enhance some differentiating characteristics between the two formulations, three initial conditions are used. Two have previously been described in Chapters V and VI. Their

main distinguishing feature is the arbitrary perturbation amplitude (APA) magnitude. The nearly linear case has an APA = 1.1 m/s compared to 5.5 m/s for the more nonlinear case. All initial conditions have a 10 m/s mean flow component.

With the nearly linear case both models behave well. The introduction of the more nonlinear initial disturbance illustrated the boundary and computational technique sensitivity. The third initial condition is the nearly linear initial field with the wave length equal to half the domain length. This has the effect of producing two waves with the domain.

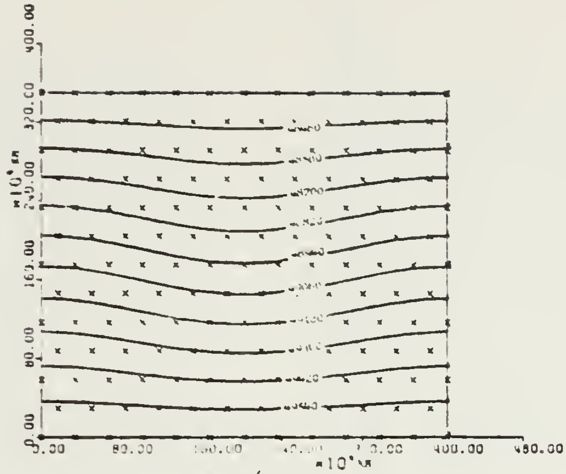
B. TEST CASE COMPARISONS

The comparisons between models are divided according to the domain geometry. The primitive model has three fields; geopotential, u and v. The vorticity-divergence model has seven fields: geopotential, stream function, velocity potential, u, v, vorticity and divergence. The geopotential, u and v will be the only fields used for the comparison.

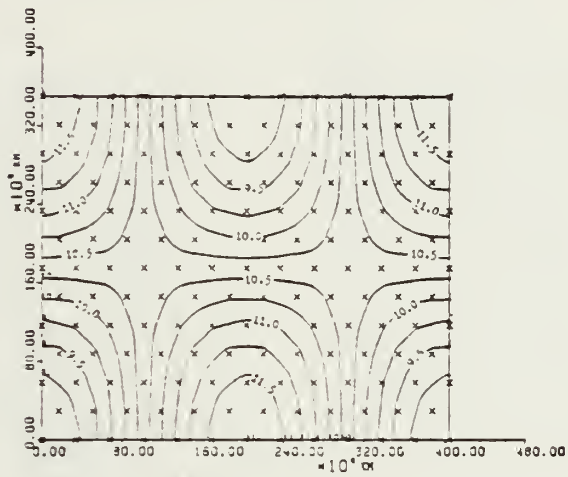
1. Regular Case

The regular domain geometry (see Figure 16a) is used for this first comparison. The initial conditions have an APA = 1.1 m/s, as shown in the contour plots in Figure 17.

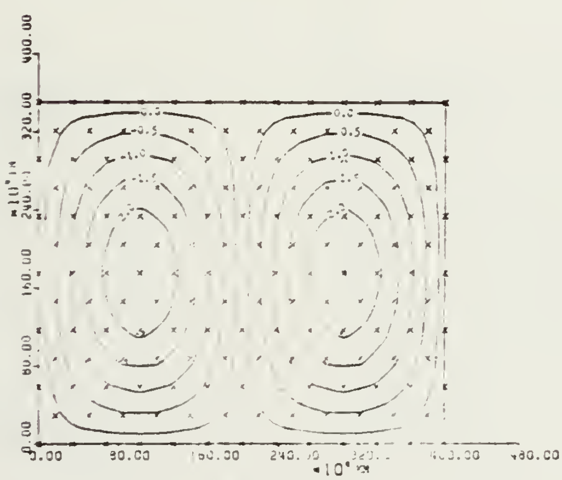
The 48 hour solutions for both models are presented in Figure 18. As anticipated, all of the solutions have



A) INITIAL Ø FIELD (GPM)



B) INITIAL U FIELD (M/S)



C) INITIAL V FIELD (M/S)

Figure 17. Initial fields for both the primitive and vorticity--divergence models using the regular domain and perturbation amplitude of 1.1 m/s.

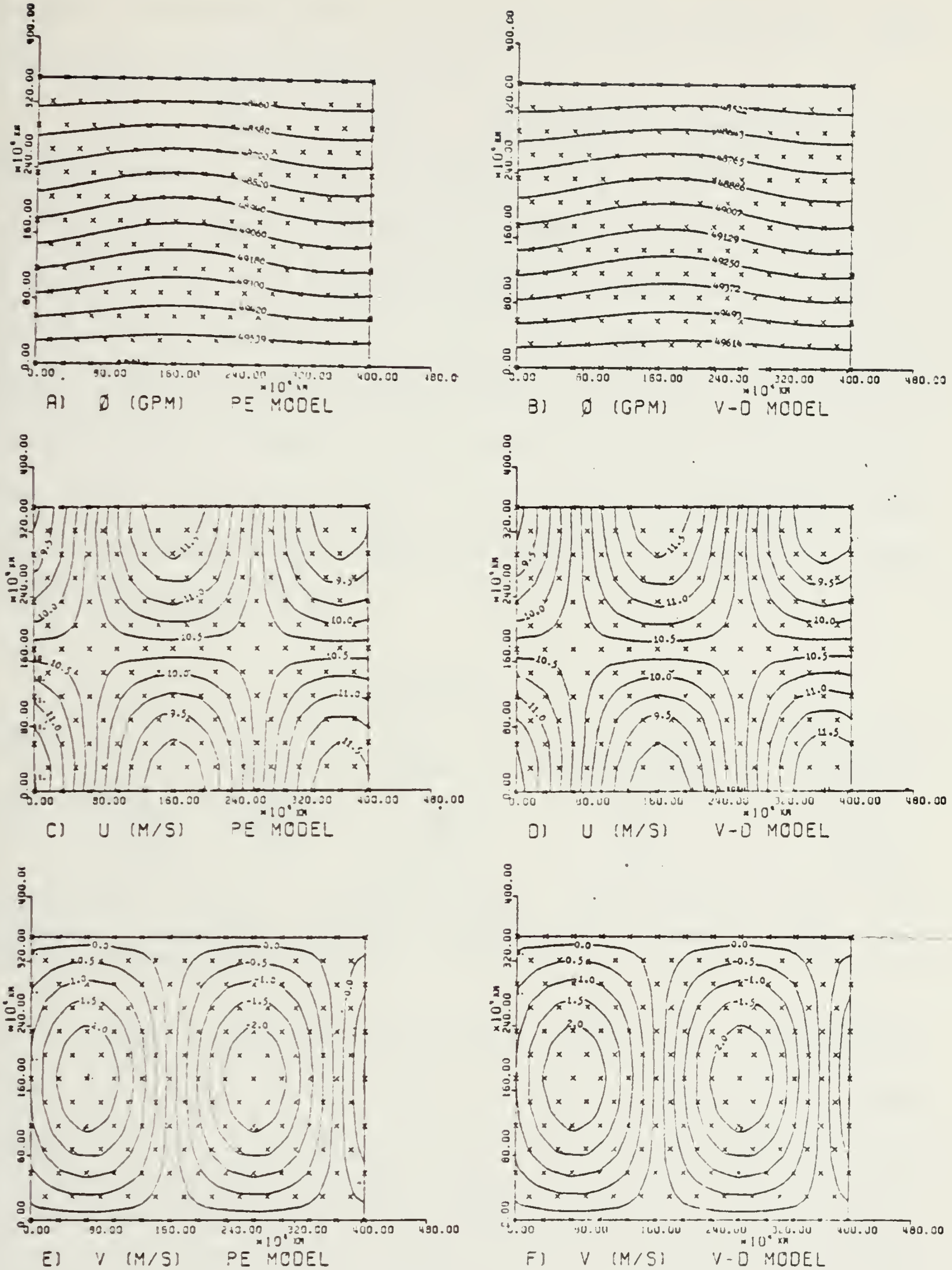


Figure 18. 48 hour forecast comparison between the primitive model and the vorticity-divergence model using the regular domain. (APA = 1.1 m/s).

smoothly contoured fields with no noticeable noise and their phase velocities are comparable.

With the nearly identical solutions, the distinguishing feature between the models is the computational time. The computational time is determined by the size of the time step each model is allowed to use as indicated on equation III-26. For this domain the grid spacing is 337 KM. The maximum phase velocity possible is different for each model. The primitive model allows gravity waves so c = 300 m/s whereas the vorticity-divergence model filters out these high frequency waves. A c = 10 m/s is used for this formulation.

Table 1

Model	Δx (KM)	c (m/s)	Δt (sec)	# of steps (iterations)	CPU units (sec)
PE	337	300	458	376	148.5
V-D	337	10	13,758	13	33.0

Comparisons of the computational times for a 48 hour forecast between the primitive model and the vorticity-divergence model.

Table 1 compares the results of the computational times for both models. The vorticity divergence model produced as accurate results over the regular domain using 22% of the computational time needed by the primitive model. In fact from geostrophic reasoning the vorticity-divergence model should produce better forecasts when small scale features are present.

2. Smooth Case

The smooth domain geometry (Figure 16b) is used in this second comparison. The initial fields shown in Figure 19 have a disturbance amplitude of 1.1 m/s.

The 48 hour solution comparison is presented in Figure 20. All fields again have smooth contoured plots. Close inspection of the two u fields, Figures 20c and d, shows that the primitive u field has small kinks along some contours and a weak tilt near the central boundary nodes, although this may be a function of the plotting routine. Notice the good symmetry for the vorticity-divergence u field.

As in the regular case, this smooth experiment produced two acceptable solutions. Again, the computational time is the differentiating criteria.

Table 2

Model	Δx (KM)	c (m/s)	Δt (sec)	# of steps (iterations)	CPU units (sec)
PE	199	300	271	638	304.8
V-D	199	10	8124	21	49.5

Comparison of the computational times for a 48 hour forecast between the primitive model and the vorticity-divergence model using the smooth domain.

Table 2 compares the computational times for both models. The vorticity-divergence has a 83.8% saving of CPU time.

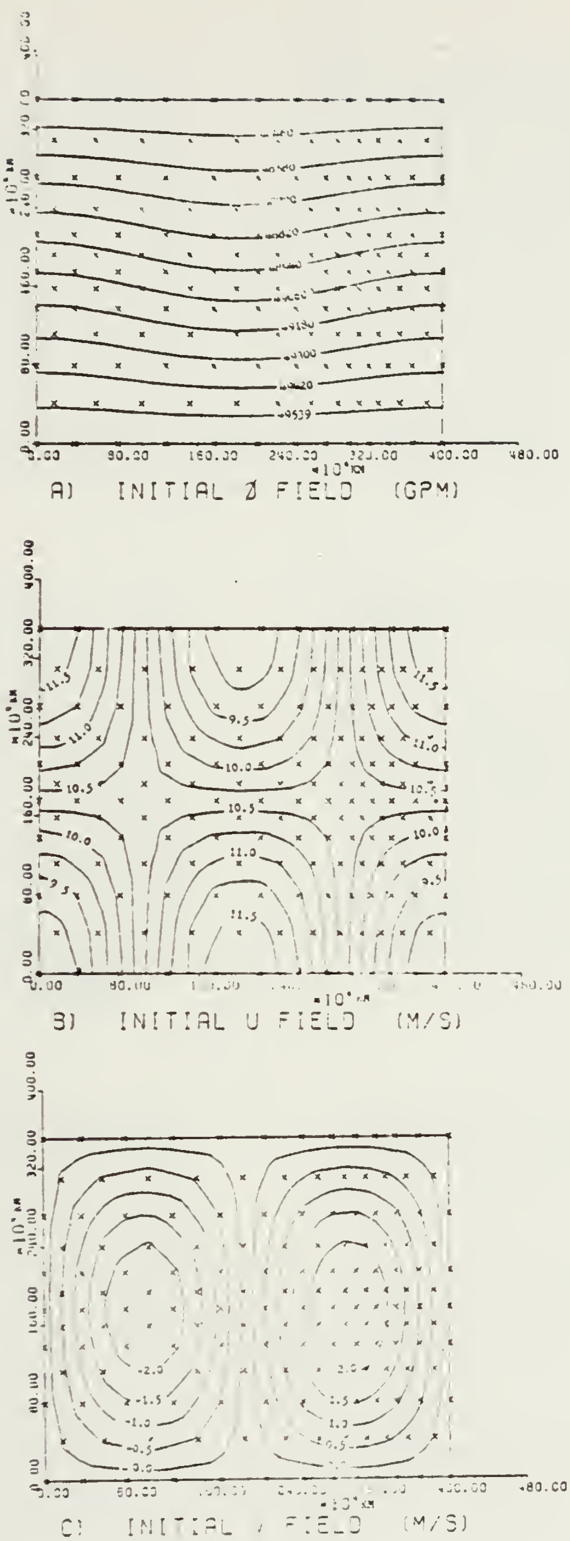


Figure 19. Initial fields for both the primitive and vorticity-divergence models using the smooth domain and perturbation amplitude of 1.1 m/s.

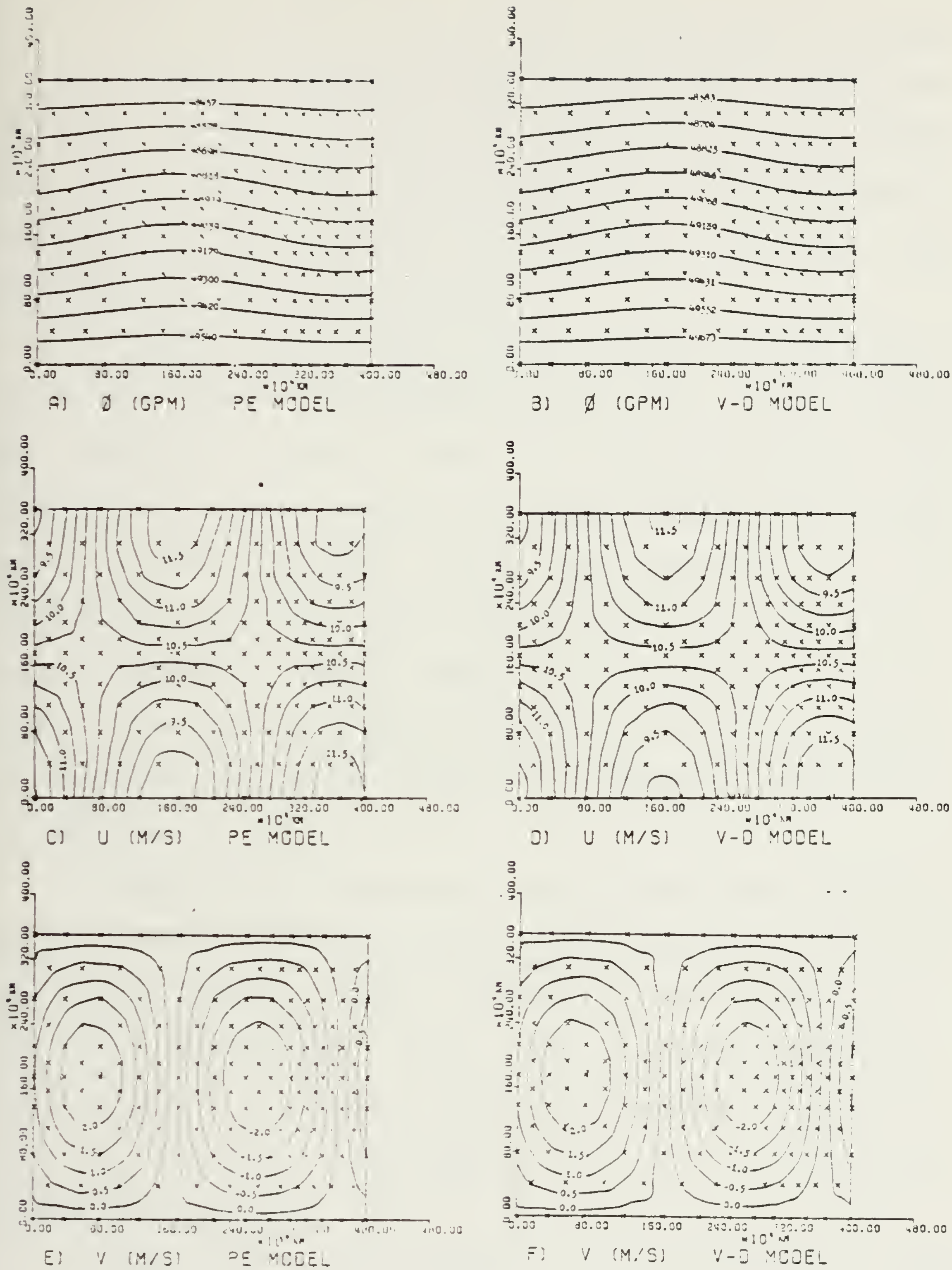


Figure 20. 48 hour forecast comparison between the primitive model and the vorticity-divergence model using the smooth domain. (APA = 1.1 m/s)

In an effort to contrast the computational accuracy of the models, this experiment is repeated using the more nonlinear initial case. Figure 21 shows the initial fields with APA = 5.5 m/s. This larger initial disturbance is reflected in the greater geopotential amplitude and magnitudes of the contour lines.

Figure 22 shows the 48 hour solution comparison. As in the more linear case presented above, all of the plots have smooth contours with no noticeable noise. The vorticity-divergence geopotential field, Figure 22b, has the ridge extending farther north, and flattening of the southernmost contour, than does the primitive geopotential field, Figure 22a. The mean geopotential heights for both models have also increased. The primitive mean geopotential is now 49080 gpm and the vorticity-divergence geopotential is 49620 gpm.

These two discrepancies indicate that the boundaries are not handled accurately. During the implementation of the vorticity-divergence model, treatment of the boundaries was the most troublesome phase. The vorticity-divergence formulation is a complex equation set and time limitations restricted further investigation of more sophisticated boundary conditions.

This same initial condition is now extended to a 96 hour forecast, which is shown in Figure 23. The mean vorticity-divergence geopotential increased to 49800 gpm,

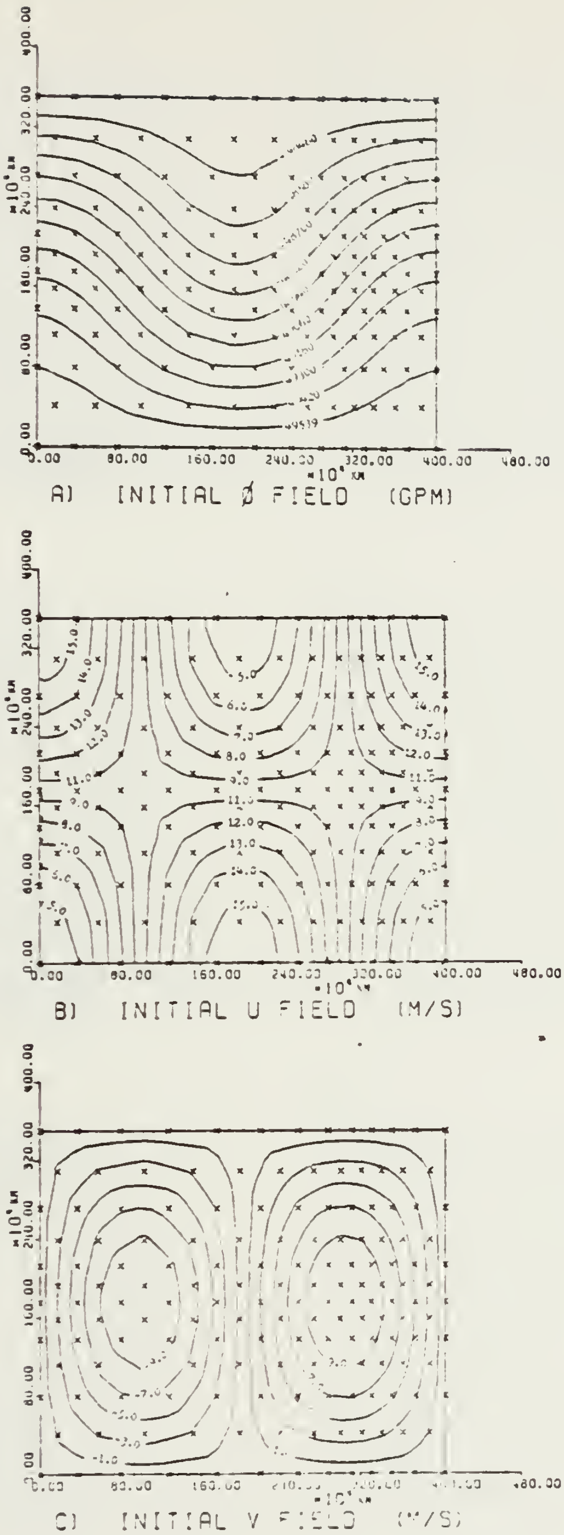


Figure 21. Initial fields for both the primitive and the vorticity-divergence models using the smooth domain and perturbation amplitude of 5.5 m/s.

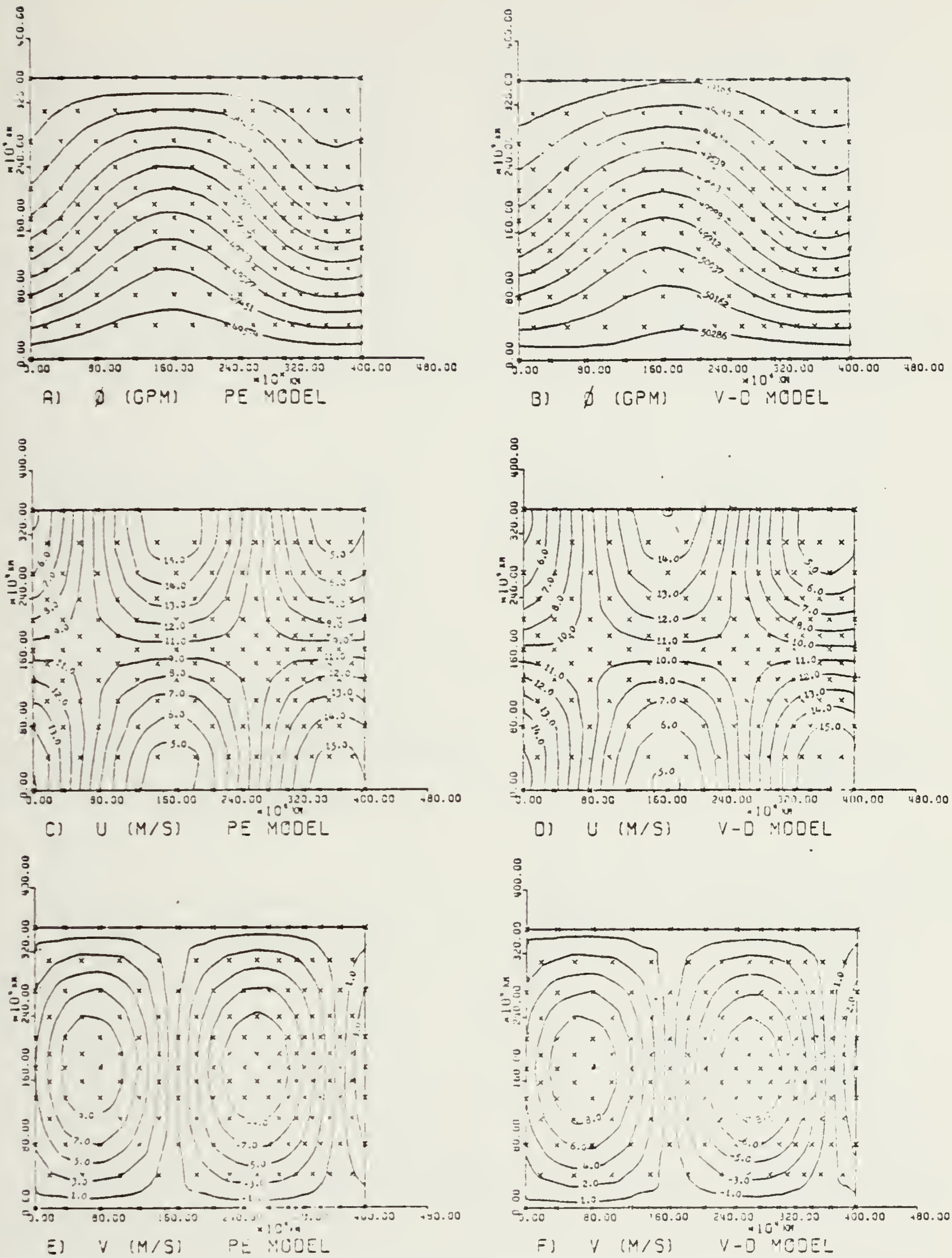


Figure 22. 48 hour forecast comparison between the primitive model and the vorticity-divergence model using the smooth domain. (APA = 5.5 m/s)

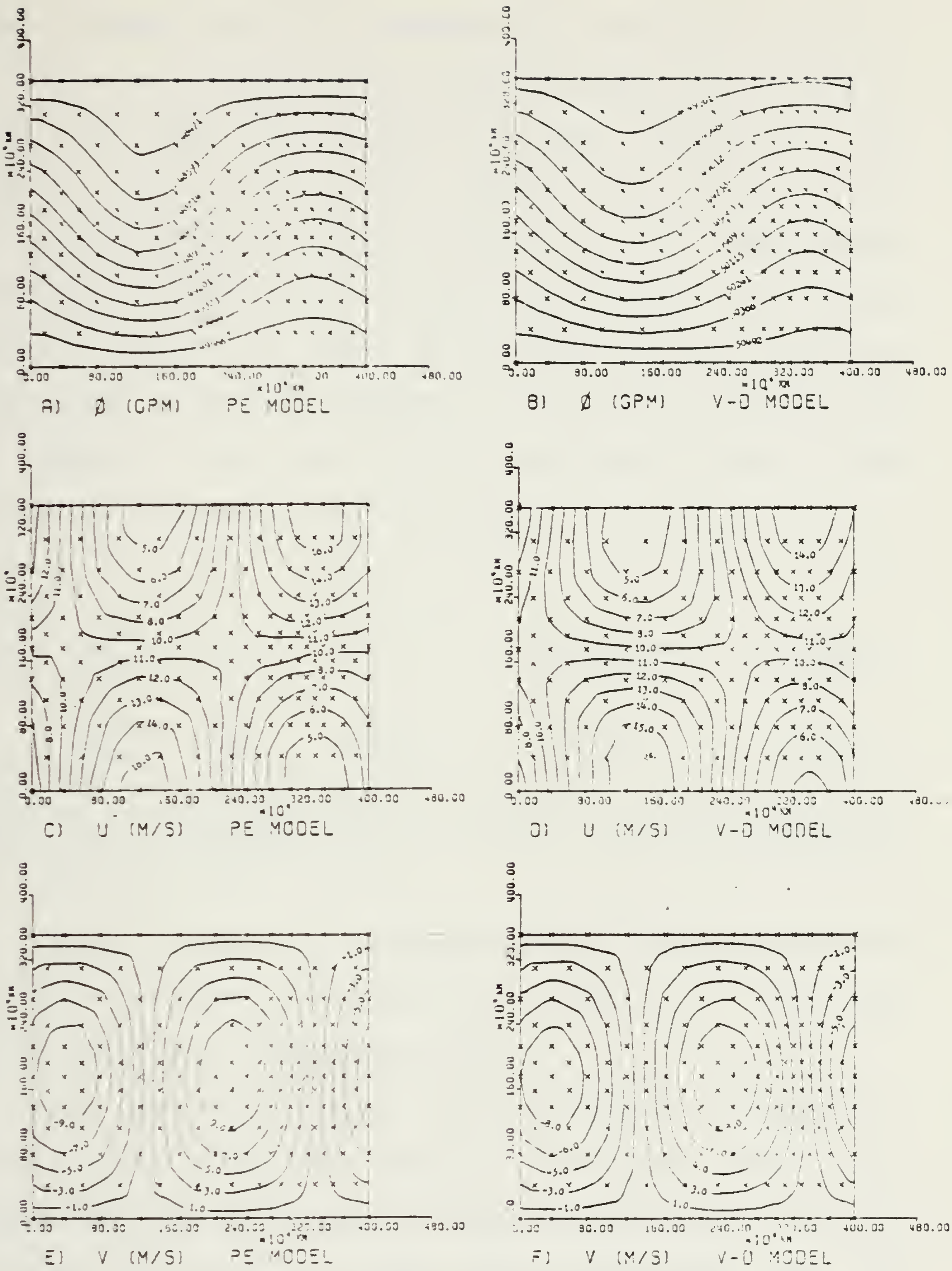


Figure 23. 96 hour forecast comparison between the primitive model and the vorticity-divergence model using the smooth domain. (APA = 5.5 m/s)

whereas the mean primitive geopotential remained constant. All fields have smooth contours. Close inspection of both u fields, Figure 23c and d, shows a slight skewing of the contours along the central channel grid points. The vorticity divergence u field has the most pronounced deviations. A hypothesis for this skewing, explained further below, is that it is caused by the computational technique employed. The relaxation schemes are extremely sensitive and fine tuning of the relaxation coefficient would have required more time than was available.

Table 3

Model	Δx (KM)	c (m/s)	Δt (sec)	# of steps (iterations)	CPU units (sec)
PE	199	320	271	1276	594.0
V-D	199	10	8124	42	91.0

Comparison of the computational times for a 96 hour forecast between the primitive model and the vorticity-divergence model using the smooth domain.

Table 3 shows the comparison between both models for the 96 hour forecast. A savings of 85 percent is realized with the vorticity divergence model.

The above experiment points out the two areas where the vorticity — divergence model is presently weak, the increase of the geopotential and the sensitivity of the relaxation coefficients. Both these weaknesses can be improved and are not a result of the formulation but of the implementation. At present, their influence is not detected

except in extremely long forecasts, such as in the 96 hour example. Further experiments may still be required if they demonstrate significant differences in the solutions of the two models.

The last experiment on the smooth domain uses the more linear initial condition APA = 1.1 m/s, but its wave length is divided by two, so that two shorter waves are propagated through the channel. The initial conditions are shown in Figure 24. Decreasing the wave length has the same effect as decreasing the density of grid points. In this case six grid points are used to describe the wave structure versus the 12 used in the previous cases.

The 48 hour comparisons are shown in Figure 25. As in all previous cases, a computational time saving of 84% is gained with the vorticity-divergence model. With fewer grid points describing the wave structure, more small-scale noise is introduced into the solution. Comparing the primitive geopotential field, Figure 25a, to the initial geopotential, Figure 24a, shows a dampening of the wave amplitude, whereas the vorticity-divergence geopotential, Figure 25b, correlated well with the initial geopotential.

The high frequency noise is evident on both u fields, Figures 25c and d. The primitive model u field is poorly defined along the boundary and becomes irregular over the interior grid points.

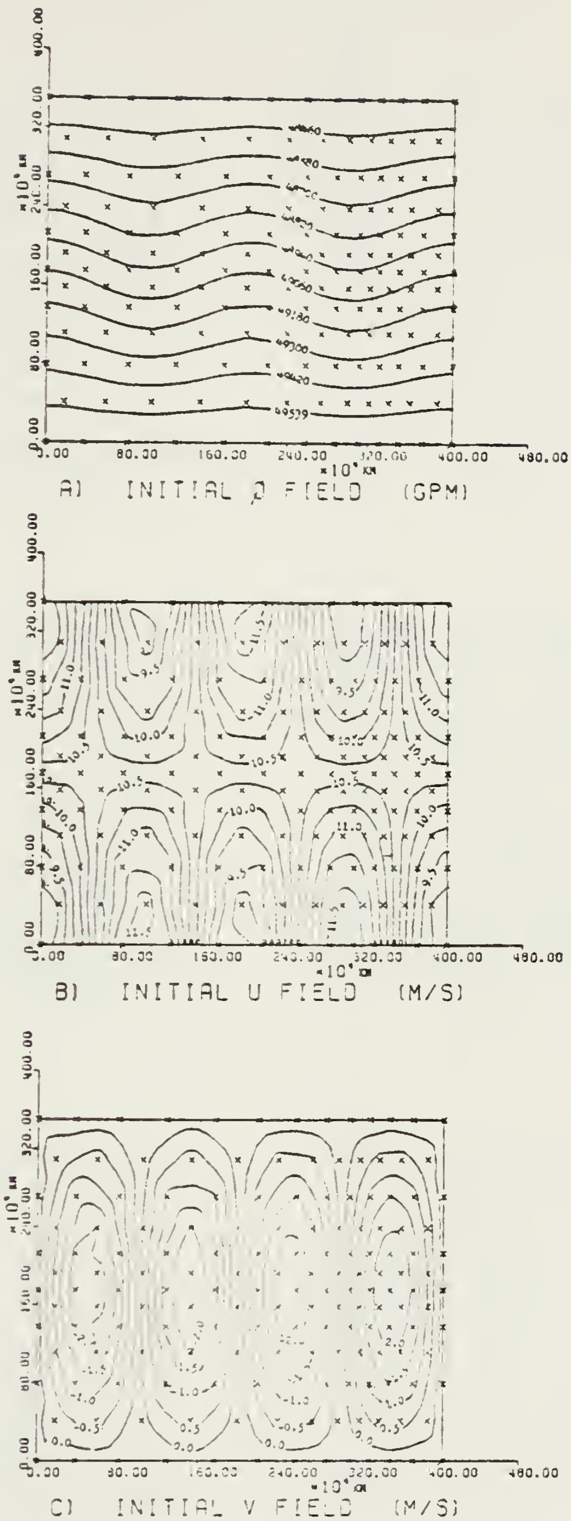


Figure 24. Initial fields for both the primitive and the vorticity-divergence models using the smooth domain. There are two waves embeded in the flow and APA = 1.1 m/s.

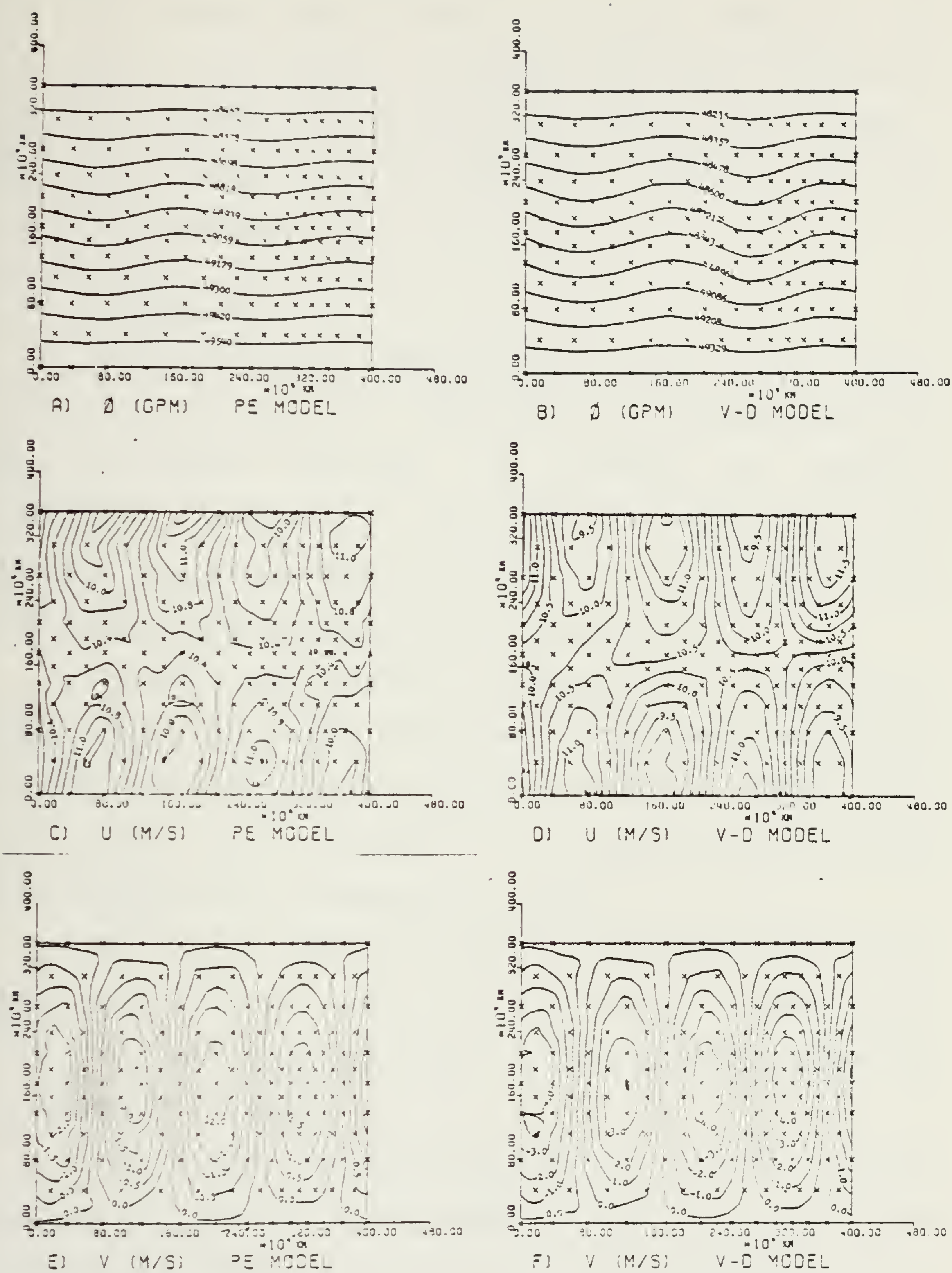


Figure 25. 48 hour forecast comparison between the primitive model and the vorticity-divergence model using the smooth domain. Two waves are embeded in the flow and APA = 1.1 m/s.

The smooth domain allows a variable resolution of grid points and produces excellent results. As the wave length gets shorter, or the forecast length gets larger, some small-scale noise is apparent, especially with the primitive formulation.

3. Abrupt Case

The abrupt case comparison uses the abrupt domain geometry (see Figure 16c). This grid point configuration is used to further illustrate the effects of spatial resolution. The previous case using the smooth domain and half wave length introduced noise into the solution, but the spatial resolution changed slowly and gradually.

Consider the transition necessary in an operational model, where the luxury of having a long smooth transition into the region of high resolution may not be possible. The abrupt domain is an example of the results obtained when spatial resolution is decreased rapidly.

The initial fields are shown in Figure 26. The more linear case, APA = 1.1 m/s, is used to eliminate effects due to the initial field, so that only the effects due to the grid geometry are seen.

The 48 hour comparisons are shown in Figure 27. Both solutions are affected by this geometry, but the primitive solutions are totally disorganized and unacceptable.

Table 4 shows the comparison of computational times for both models for a 48 hour forecast. An 85% savings in

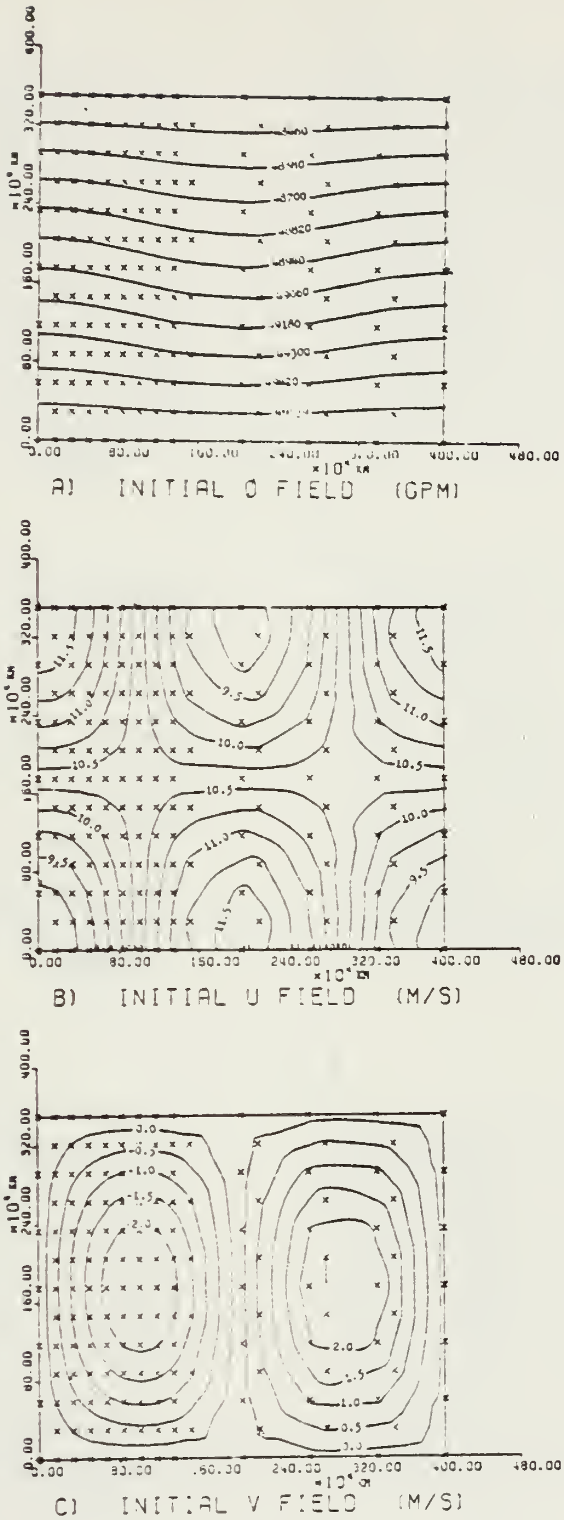


Figure 26. Initial fields for both the primitive and vorticity-divergence models using the abrupt domain and perturbation amplitude of 1.1 m/s. Note that the element shapes are not equilateral triangles.

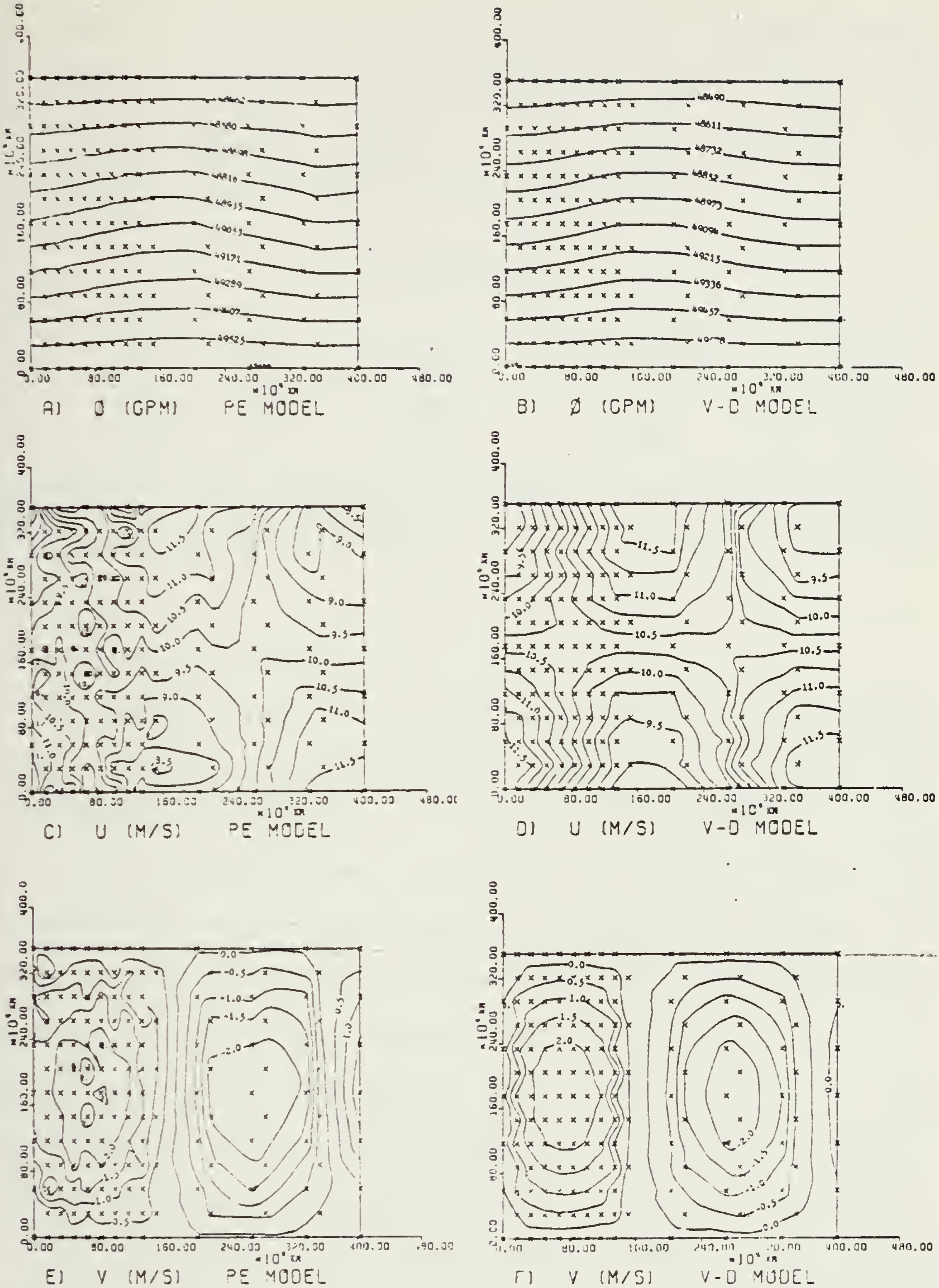


Figure 27. 48 hour forecast comparison between the primitive model and the vorticity-divergence model using the abrupt domain. (APA = 1.1 m/s).

CPU time is obtained using the vorticity-divergence model. Not only is there a computational savings with the vorticity-divergence model, but the solution is significantly better than the primitive model.

Table 4

Model	Δx (KM)	c (m/s)	Δt (sec)	# of steps (iterations)	CPU units (sec)
PE	168	300	228	756	368.9
V-D	168	10	6858	25	55.7

Comparison of the computaional times for a 48 hour forecast between the primitive model and the vorticity-divergence model using the abrupt domain.

C. COMPUTATIONAL SENSITIVITY

This section will offer an explanation for the skewing of the contours in the 96 hour forecast solution over the smooth domain (Figure 23). As mentioned previously, there was not enough time available for fine tuning the overrelaxation coefficient, which is used while solving the system of equations. The overrelaxation coefficient may be very sensitive and small changes can, on occasion, radically change the rate of convergence. The optimal value of the overrelaxation coefficient depends on the specific form of the coefficients of the equation and the error distribution.

The equation set to be solved consists of three equations and each equation required its own relaxation coefficient. When solving the equations over the regular domain, the entire system is well behaved and an optimum

relaxation coefficient is easily determined. However, as the domain geometry changes, the system of equations do not converge as rapidly and the relaxation coefficient requires further fine tuning.

The 96 hour forecast uses the smooth domain. The mid-latitudinal grid points are compacted, creates a denser belt in the middle of the channel. The coefficients originally computed using the regular domain need adjustments to properly solve the equations.

To illustrate the significance for fine tuning the relaxation coefficient, consider the series of plots in Figure 28. The vorticity divergence equation set can be simplified by assuming the flow is non-divergent, so that only the vorticity equation needs to be solved. Figure 28a is the 48 hour vorticity field using this equation over the regular domain with an overrelaxation coefficient of 1.3. The field is well defined with smooth contours.

Figure 28b is similar to the case in Figure 28a except that the smooth domain is used. Notice the V-shaped kink in the pattern with a steeper slope in the upper half. This increased bias in the upper half is caused by relaxing the field in the same direction during each pass over the domain. When the direction is reversed after each pass, the exaggerated bias in the upper half disappears, as is seen in Figure 28c. However, the V-shaped kink is not eliminated.

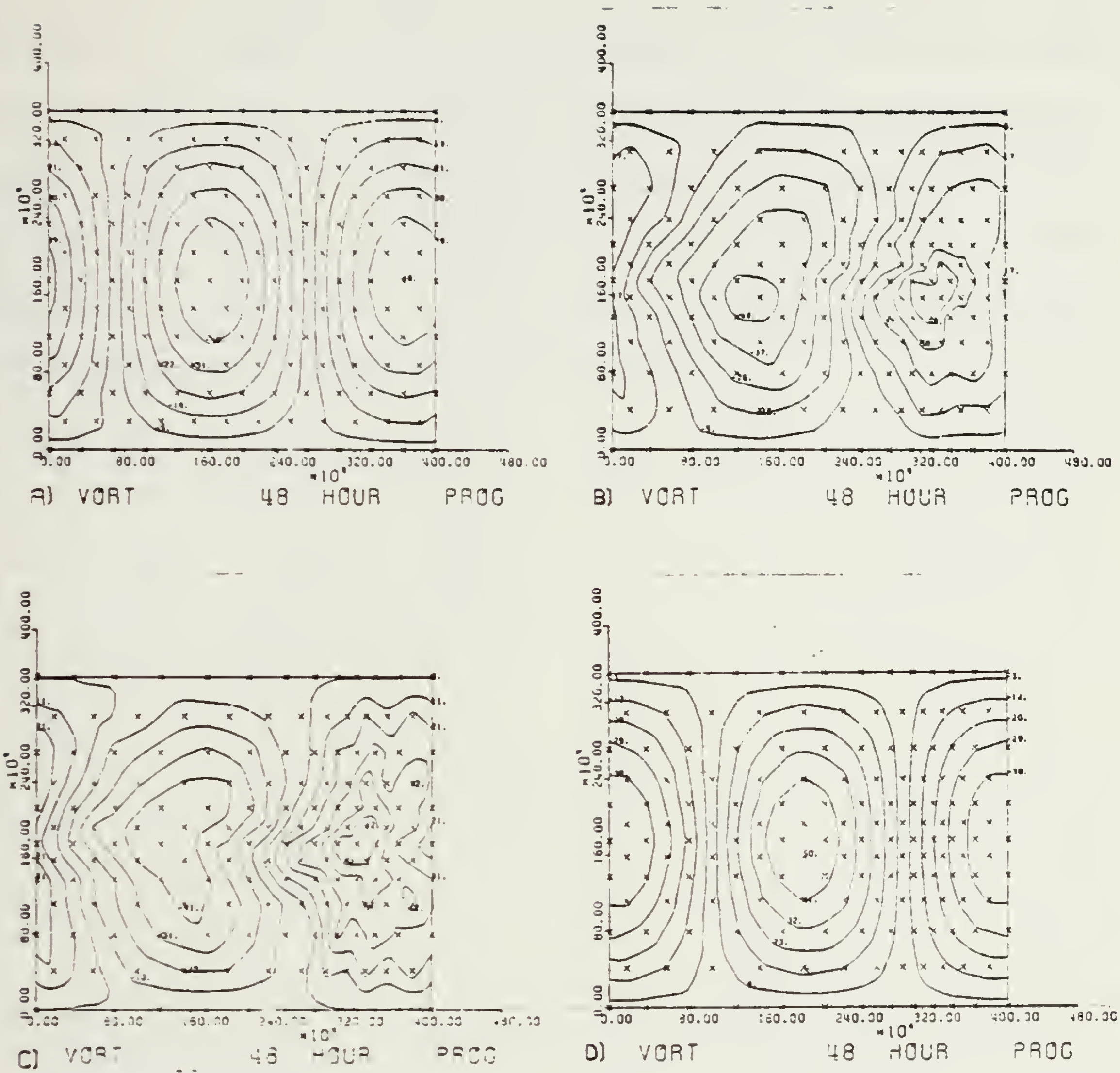


Figure 28. Computational sensitivity using the 48 hour vorticity fields. Fig. 28 A) the 48 hour forecast using the regular domain, 28 B) the same forecast using the smooth domain relaxing in one direction, 28 C) same as 28 B) except alternate the direction of relaxation after each pass over the domain, and 28 D) same as 28 C) except the relaxation coefficient was fine tuned form 1.3 to 1.297.

Varying the relaxation coefficient from 1.3 to 1.297 produces the much improved solution in Figure 28d. If the relaxation coefficients for the other two equations could also be fine tuned, it appears that improved solutions would result. There are also other relaxation techniques available that have potential for improving the solution while also converging at a faster rate. Some of these techniques will be tested in the futuer using this model.

VIII. CONCLUSIONS

This research investigated different finite element formulations for the shallow water equations. The two-dimensional domain was a channel which simulated a belt around the earth. Analytic initial conditions were used to simplify the comparisons. Two formulations were examined; one using different shaped basis functions and the other using a different form of the equation. Each was compared to the primitive form of the shallow water equations that was developed by Kelley (1976).

The use of equilateral shaped elements which was suggested by Dr. M.J.P. Cullen significantly improved the solutions compared to Kelley's model, which originally used right triangles as basis functions. Most of the other studies in this thesis used the equilateral triangles.

Williams and Zienkiewicz (1981) suggested the use of piecewise linear basis functions for the velocity field and piecewise constant functions for the height field. This formulation was tested with a linearized continuity equation. The results were poorer than those obtained with Kelley's model.

Most of the effort in this thesis was devoted to implementating and testing a vorticity-divergence model

similar to the ones developed by Staniforth and Mitchell (1977) and Cullen and Hall (1979). Several tests were presented which compare this formulation with Kelley's model. It was found that this model executes approximately one order of magnitude faster than does the primitive formulation used by Kelley. Secondly, as the spatial resolution between grid points decreases, this formulation produces a solution that is far superior to the primitive form. A disadvantage is its computational sensitivity, which requires fine tuning in solving the elliptic equations for certain geometries. It also requires 25 percent more computer storage, due to the more complex equation set and the additional variables that are treated.

Implementation of finite element models is not easy. However, there is a lot of generality and redundancy imbedded in the computations. Versatile modules were written which significantly reduced the effort in implementing the vorticity-divergence model.

Further research is suggested using this finite element formulation. It has accurate phase propagation, is able to handle variable grid geometry, reduce the small-scale noise and decrease the model's execution time. Specifically more advanced methods of solving the elliptic equations should be investigated. Finally, the formulation should be tested with small-scale forcing, where its advantages should be most evident.

APPENDIX A

A. DEFINITIONS

[illegible]

AMP......AMPLITUDE OF INITIAL FIELD

DELTAY...GRID POINT SPACING IN Y AXIS (METERS)
DELTAX...GRID POINT SPACING IN X AXIS (METERS)
DIVDIF...DIFFUSION COEF FOR DIVERGENCE EQUATION
DT.......TIME INTERVAL FOR TIME INTEGRATION (HOURS)

E........RADIUS OF THE EARTH (METERS)

FCSTHR...MAX FORECAST HOUR
F0.......CORIOLIS VALUE AT THE MID CHANNEL LATITUDE

HOUR.....CURRENT FORECAST TIME (HOURS)

IFLD.....TEST FIELD USED FOR INITIALIZATION (PHI,PSI,CHI)

LEAPFR...NUMBER OF ITERATIONS NEEDED TO COMPLETE FCST

N........POINTER TO THE NTH TIME LEVEL IN FCST ARRAY
NLAT.....NUMBER OF GRID POINTS ON THE Y AXIS
NLON.....NUMBER OF GRID POINTS ON THE X AXIS
NM1......POINTER TO THE (N-1) TIME LEVEL
NN.......SIZE OF THE COMPACTED CONNECTIVITY ARRAY
NOELMT...TOTAL NUMBER OF ELEMENTS IN THE DOMAIN
NONODE...TOTAL NUMBER OF NODES IN THE DOMAIN
NOPT.....NUMBER OF INTERPOLATION POINTS
NP1......POINTER TO THE (N+1) TIME LEVEL

PHIAVG...AVERAGE PHI OVER THE INITIAL PHI DOMAIN
PHIDIF...DIFFUSION COEF FOR CONTINUITY EQUATION

RUNCDE...RUN CODE NUMBER TO IDENTIFY EACH RUN
R1.......RATIO TO VARY X AXIS BY
R2.......RATIO TO VARY Y AXIS BY

SRCCOF...SOURCE FUNCTION COEFFICIENT

THETA....LATITUDE OF SOUTHERN BOUNDARY
THETAO...LATITUDE OF MID CHANNEL
THETAN...LATITUDE OF NORTHERN BOUNDARY

UBAR.....MEAN HORIZONTAL WIND

W........WIDTH OF THE CHANNEL (METERS)
WAVES....TOTAL NUMBER OF WAVE LENGTHS IN CHANNEL

XL.......LENGTH OF THE CHANNEL (METERS)
XLOFSET..OFFSET, MOVES CHANNEL NORTH OR SOUTH [illegible]
XMIN.....MIN X VALUE FOR DOMAIN

YSOUTH...DISTANCE TO SOUTHERN BOUNDARY FROM EQUATOR
YNORTH...DISTANCE TO NORTHERN BOUNDARY FROM EQUATOR

YMIN.....MIN Y VALUE FOR THE DOMAIN

ZETDIF...DIFFUSION COEF FOR VORTICITY EQUATION

[illegible] IDENTIFICATIONS

A........COEFF. FOR TRANSFORMATIONS TO AREA COORD.
AREA.....[illegible]
[illegible]
[illegible]
[illegible]

B........COEFF. FOR TRANSFORMATION TO AREA COORD.
B2.......MATRIX USED FOR ASSEMBLING MASS MATRICES
BETA.....[illegible]

C........[illegible] VECTOR FOR CONSTRUCTION OF MASS MATRICES
CHI......VELOCITY POTENTIAL FIELD

DIV......DIVERGENCE FIELD

ELNODE...IDENTIFIES THE NODES FOR EACH ELEMENT

F........MASS MATRIX USED IN SOLVING CONTINUITY EQUATION

G........MASS MATRIX USED IN SOLVING DEPENDENT VARIABLES

H........MASS MATRIX USED IN SOLVING VORT & DIV EQ.

IBD......IDENTIFIES THE BOUNDARY NODES
IPRNT....FLAG FOR OUTPUT SELECTIONS
ISTART...ELEMENT STARTING LOCATION IN CONNECTIVITY

[illegible].....CONNECTIVITY MATRIX
NUM......TOTAL NUMBER OF NODES CONNECTED TO EACH ELEMENT
NTRI.....NUMBER OF TRIANGLES INTERPOLATED

PAST.....PREVIOUSLY SOLVED PHI, PSI & CHI FIELDS
PHI......GEOPOTENTIAL FIELD (DEPARTURE FROM MEAN)
PSI......STREAM FUNCTION FIELD

QU.......U*(ABSOLUTE VORTICITY)FIELD
QV.......V*(ABSOLUTE VORTICITY)FIELD

RHS......RHS VECTOR USED IN CALCULATING THE RIGHT
HAND SIDE (FORCING TERM) OF ALL EQUATIONS

SAVE.....STORE RHS VALUES FROM VORTEQ AND DIVEQ
SOR......SOURCE FUNCTION VALUES

U........U COMPONENT OF MOTION FIELD
UM1......U(N-1) FIELD

V........V COMPONENT OF MOTION FIELD
VIJ......INNER PRODUCT <VJ,VI>
VIJK.....INNER PRODUCT <VJ,VK,VI>
VJK......INNER PRODUCT <VJX,VI>
[illegible]....INNER PRODUCT <VJY,VI[illegible]>
[illegible]....INNER PRODUCT <VJ[illegible],VI[illegible]>
[illegible]....INNER PRODUCT <VJ[illegible],VI[illegible]>
[illegible]....INNER PRODUCT <VJ[illegible],VI[illegible]>
[illegible]....INNER PRODUCT <VJ[illegible],VI[illegible]>

KE.......KINETIC ENERGY FIELD
XLONG....NODE'S DISTANCE FROM ORIGIN ALONG X AXIS
XI.......NEW INTERPOLATED X COORDINATES

YLAT.....NODE'S DISTANCE FROM ORIGIN ALONG Y AXIS
YI.......NEW INTERPOLATED Y COORDINATES

ZETA.....VORTICITY FIELD

SUBROUTINE DEFINITIONS

END

J. MAIN PROGRAM

```
      DIMENSION [illegible] T,[illegible],ISTART,NAME,I[illegible])
      [illegible]
      COMMON [illegible], [illegible], [illegible], [illegible], EPLS
      COMMON [illegible]/[illegible],[illegible],[illegible],[illegible],[illegible],DT
      COMMON [illegible]/[illegible],T([illegible]),[illegible],DIF,[illegible],[illegible])F
      COMMON [illegible]/[illegible](1536),[illegible](1536),[illegible](1[illegible]+4)
      COMMON [illegible]/[illegible](1536),[illegible](1536),[illegible](1[illegible]),P[illegible](1[illegible],3)
      COMMON [illegible]/[illegible](1536),[illegible],[illegible](1536),XI(12),YI(1[illegible])
      COMMON [illegible]/[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],F)
      COMMON [illegible]/[illegible](1536),[illegible](1536),[illegible],SPC[illegible]F,S[illegible](1536)
      COMMON [illegible]/[illegible](3,2[illegible]),[illegible](3,2[illegible]),[illegible](2[illegible])
      COMMON [illegible]/[illegible](3,2[illegible]),[illegible](3,2[illegible]),/IJ[illegible](3,3,3,2[illegible])
      COMMON [illegible]/[illegible](2[illegible]),/IJ[illegible](3,3,3,2[illegible]),/IJ[illegible](3,3,3,2[illegible])
      COMMON [illegible]/[illegible](1536),[illegible](1536),[illegible](1536),[illegible](1536),[illegible](1[illegible])
      COMMON [illegible]/[illegible](1536,3),[illegible](1536,3),[illegible](1536,3),[illegible](1536,[illegible])
      COMMON [illegible]/[illegible](1[illegible]+4),[illegible](1[illegible]+4),[illegible](1[illegible]+4)
      COMMON [illegible]/[illegible](3,3),[illegible](1[illegible]+4)
      COMMON [illegible]/[illegible](3,3,2[illegible]),/JY[illegible](3,3,2[illegible])
      COMMON [illegible]/[illegible]/[illegible](3,3,2[illegible]),/J[illegible](3,3,2[illegible])
      COMMON [illegible]/[illegible](1536),[illegible](1536),[illegible](1536),BETA(1536),[illegible](1536)
      COMMON [illegible]/[illegible],[illegible],FCSTHR
      COMMON [illegible]/IPRINT/IPRINT(20),[illegible],[illegible],[illegible],[illegible],[illegible]
C
C
C
C     INITIALIZATION   PHASE
C
C
C
C     INITIALIZE MODEL OPTION VARIABLES
      CALL INITO
C
C     INITIALIZE MODEL PARAMETERS
      CALL INITPF
C
C     INITIALIZE INPUT FIELDS PSI, CHI AND PHI
      CALL IC
C
C     INITIALIZE ALL DEPENDENT VARIABLES
      CALL DEPVA
C
C
C     FORECAST   PHASE
C
C
C     USE MATSUNO TECHNIQUE FOR FIRST FORWARD STEP
      CALL MATSNO (DT)
C
C     USE LEAPFROG FOR REMAINDER OF FCST PERIOD
      CALL LEAPF [illegible]
C
C
      STOP
      END
```

D. [illegible] PHASE

```
      SUBROUTINE [illegible]
C
C  [illegible] SOLUTION.
C
      [illegible]
      [illegible]
     1  [illegible],PHI[illegible]
     2  [illegible]
     3  [illegible]/[illegible]T([illegible]),PHI[illegible]
     4  [illegible]
     5  [illegible](150),YLAT(150),AMP,[illegible](150)
     6  [illegible],FCST[illegible]
     7  [illegible]T(20)),[illegible]
C
      [illegible] (IF [illegible] UPDATED)
      [illegible]=1
      [illegible]=1.0
      [illegible]=[illegible] LAT
      [illegible]=[illegible]*( LAT+1)
      [illegible]=[illegible]-2*([illegible])
C
C     [illegible]
      CALL [illegible] ([illegible])
C
C     [illegible] IS 1.0 TO [illegible])
      [illegible] = 1.0
C
C     GRID ASPECT RATIO: [illegible] => X AXIS, [illegible] => Y AXIS.
C     [illegible] 1.0 TO 4.0 (CAN NOT BE LESS THAN 1.0)
C     [illegible] GRID USE R1 = -1.0
      R1 = [illegible]
      R2 = 2.5
C
C     DIFFUSION COEF. [illegible] TO 0.05
C     NOTE: [illegible] MAY HAVE TO BE DECREASED [illegible] STABILITY
      [illegible] = [illegible]
      [illegible] = [illegible]
      [illegible] = [illegible]
C
C     FCST ITERATION NUMBER. I.E. 15 ITERATIONS = 43 HR FCST
C     [illegible] TRANSFORMED LENGTH COMPUTED [illegible], 43 HR
C     [illegible]; [illegible]; 7=24; 11=36; 15=48; 19=60; 21=72;
      [illegible] = 15
C
C     CHANNEL LATITUDE BOUNDARIES
      [illegible] = [illegible]
      [illegible] = 33.0
      [illegible] = [illegible]
      [illegible]=6.37[illegible]
C
C     [illegible] FLOW
      [illegible] = 1[illegible]
C
C     [illegible] SOURCE [illegible]
C     [illegible]; [illegible] = 0.0 IF [illegible]
      [illegible] = [illegible]
      [illegible] = [illegible]/5.7
      [illegible] = [illegible]/1.2
      [illegible] = [illegible]
C     [illegible] = 0.0 IF NO SOURCE
      [illegible] = [illegible]
C
      [illegible] = [illegible]
      [illegible] = [illegible]
      [illegible] = [illegible]
      [illegible] = [illegible]
      [illegible] = [illegible]
      [illegible] = [illegible]
```

```
C     SETS THE FLAG FOR THE OUTPUT ROUTINE: 1 = ON, 0 = OFF
C     PRINTS [illegible] CONNECTIVITY MATRIX
      IPRNT(1) = 0
C     [illegible]
      IPRNT(2) =
C     [illegible] COMPUTATED [illegible]
C     X,Y COORDINATES FOR THE GRID POINTS [illegible] CHANNEL DATA
      IPRNT(3) = 0
C     [illegible] AFTER TRANSFORMATION
      IPRNT(4) = 0
C     [illegible] PRODUCTS USED IN THE INTEGRATIONS
      IPRNT(5) = 0
C     INITIAL FIELDS PHI, PSI, CHI
      IPRNT(7) = 0
C     INITIAL U,V,KE,VORT,DIV,UU,VV,ALFA,BETA FIELDS
      IPRNT(8) = 0
C     SUPPLEMENT FIELDS AFTER EACH ITERATION IN SOLVER
      IPRNT(9) = 0
C     ANALYTICAL VALUES FOR U,V,KE,VORT,DIV,UU,VV,ALFA,BETA
      IPRNT(10)= 0
C     PHI, PSI AND CHI FOR PRINT TIME INTERVAL IN IPRNT(12)
      IPRNT(11)= 1
C     PRINT INTERVAL IN HOURS (MUST BE SOME MULTIPLE OF DT)
      IPRNT(12)= 48
C     U AND V FIELDS FOR PRINT TIME INTERVAL IN IPRNT(12)
      IPRNT(13)= 0
C     U,V,KE,VORT,DIV,UU,VV,ALFA,BETA FIELDS AS IN IPRNT(13)
      IPRNT(14)= 1
C     TURNS ON THE PLOT FLAG
      IPRNT(15)= 1
C     INITIAL FIELD PLOTS
      IPRNT(19)= 0
C     SPECIFY WHICH FIELDS TO PLOT;7 DIGIT BINARY NUMBER
C     1/0 IN RESPECTIVE POSITION :PHI:PSI:CHI:U:V:VORT:DIV:
C     TO PLOT PSI,U,VORT FIELDS: IPRT(16) = 0101010
      IPRNT(16)= 1001100
C     HARMONIC ANALYSIS
      IPRNT(17) = 1
C     FIELD ==> :PHI:PSI:CHI:U:V:VORT:DIV:
      IPRNT(18) = 1000000
      RETURN
      END
```

```
SU[illegible] [illegible] [illegible] ([illegible]Y,[illegible])
[illegible] [illegible]Y, [illegible]
[illegible] = [illegible]
[illegible]
```

```
      SUBROUTINE INITFE
C
C  INITFE - INITIALIZE THE MODEL PARAMETERS. THIS INCLUDES
C  GENERAL PARAMETERS, SETTING UP A BOUNDARY [illegible] USED FOR
C  [illegible] TEST [illegible] COMPUTING [illegible] FINITE ELEMENT
C  [illegible]
C  [illegible]
C  [illegible] FINITE ELEMENT.
C
C     BASIC FINITE ELEMENT ROUTINES
C
C     DIVIDE DOMAIN INTO ELEMENTS AND NUMBER THE NODES
      CALL [illegible]
C
C     CONSTRUCT THE CONNECTIVITY MATRIX
      CALL [illegible]
C
C     SET ARRAY [illegible] FOR BOUNDARY TEST
      CALL SET[illegible]
C
C     COMPUTE CHANNEL CHARACTERISTICS
      CALL CHANNL
C
C     COMPUTE X, Y COORDINATES FOR THE NODES
      CALL LOCATE
C
C     TRANSFORMS THE UNIFORM GRID TO VARYING GRID
      CALL TRANS
C
C     TRANSFORMS THE UNIFORM GRID TO ABRUPT GRID MESH
      CALL ABRUPT
C
C     TRANSFORM CARTESIAN COORD TO AREA COORD
      CALL AREA
C
C     COMPUTE ALL INNER PRODUCTS
      CALL INNER
C
C     CALCULATE MASS MATRIX
      CALL AMTRXI
C
C     SET UP PARAMETERS NEEDED FOR INTERPOLATION
      CALL [illegible]SET
C
      RETURN
      END
```

```
      SUBROUTINE NUMERS
C
C     NUMERS - ASSIGNS TO EACH ELEMENT THE GRID NUMBER (NODE)
C     ASSOCIATED WITH EACH NODE, FOR ALL NODES PER ELEMENT.
C
      [illegible]
      [illegible]
     1 [illegible]
     2 [illegible]
     3 [illegible]
     4 [illegible]
     5 [illegible]
C
      DATA NODE1/1/, NODE2/2/, NODE3/3/, ELMNBR /0/
      DATA ODD/1/,EVEN /2/,ROW/1/,LWRLFT/0/
C
C     FIND THE [illegible] WITH ELEMENT [illegible] NUMBERS
      DO 20 [illegible] = 1, NLAT
      IF (ROW.EQ.EVEN ) LWRLFT = LWRRHT - 1
      DO 10 NT=1, NLONG
C
C     CALCULATE 4 NODE NUMBERS TO BE USED IN THE ASSIGNMENT
      LWRLFT = LWRLFT + 1
      LWRRHT = LWRLFT + 1
      UPRLFT = LWRLFT + NLONG
      UPRRHT = UPRLFT + 1
C
      ELMNBR = ELMNBR + 1
      ELMENT(ELMNBR,NODE1) = LWRLFT
      ELMENT(ELMNBR, NODE2) = LWRRHT
      ELMENT(ELMNBR, NODE3) = UPRLFT
      IF (ROW.EQ.EVEN) ELMENT(ELMNBR,NODE2) = UPRRHT
C
      ELMNBR = ELMNBR + 1
      ELMENT(ELMNBR, NODE1) = LWRRHT
      ELMENT(ELMNBR, NODE2) = UPRRHT
      ELMENT(ELMNBR, NODE3) = UPRLFT
C
      IF (ROW.EQ.ODD) GO TO 10
      ELMENT(ELMNBR, NODE1) = LWRLFT
      ELMENT(ELMNBR, NODE2) = LWRRHT
      ELMENT(ELMNBR, NODE3) = UPRRHT
   10 CONTINUE
C
C     CORRECT NODE NUMBER FOR END ELEMENT; CYCLIC CONTINUITY
      LAST = ELMNBR - 1
      ELMENT(LAST,NODE2) = ELMENT(LAST,NODE2) - NLONG
      IF(ROW.EQ.ODD)ELMENT(ELMNBR,NODE1)=ELMENT(LAST,NODE2)
      ELMENT(ELMNBR,NODE2) = ELMENT(ELMNBR,NODE2) - NLONG
      IF(ROW.EQ.EVEN)ELMENT(ELMNBR,NODE3)=ELMENT(LAST,NODE2)
C
      ROW = ROW + 1
      IF (ROW.GT.2) ROW = 1
   20 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE CONREL
C
C     CONREL - CONSTRUCTS THE NODE CONNECTIVITY MATRIX IN ARRAY
C     MAME THEN PACKS IT INTO NAME. ISTART CONTAINS STARTING
C     LOCATION FOR EACH NODE'S CONNECTIVITY. NUM STORES
C     THE NUMBER OF THE CONNECTIVITY NODES FOR EACH NODE.
C
      INTEGER*2 ELEMENT,NUM,ISTART,NAME,MAME
      REAL*8 NUMBER
      INTEGER ELMNBR,NODE1,NODE2,NODSUB,NODCHK,COL,INDEX
      COMMON/CM1/NONODE,NOELMT,NN,HOUR,PHISAP,NM1,N,NP1,IFLG
      COMMON/CM2/ELEMENT(283,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/CM3/NUM(156),ISTART(156),NAME(1044)
      COMMON/IPRINT/IPRNT(20),NAVEND,R1,R2,LEAPHR,RUNNBR
      DIMENSION MAME(156,7)
C
C     INITIALIZE NUM TO 1, MAME TO 0 AND CONSECUTIVELY
C     NUMBER FIRST COLUMN OF MAME.
      DO 10 NODE = 1, NONODE
      NUM(NODE) = 1
      MAME(NODE,1) = NODE
C
C     ZERO THE REMAINING COLUMNS OF MAME
      DO 10 COL = 2, 7
   10 MAME(NODE,COL) = 0
C
C     CONSTRUCT THE CONNECTIVITY MATRIX MAME
      DO 50 ELMNBR = 1, NOELMT
      DO 50 NODSUB = 1, 3
      NODE1 = ELEMENT(ELMNBR,NODSUB)
C
C     CHECK NODE SUBSCRIPTS; IF EQUAL LOOP, ELSE CONTINUE
      DO 50 NODCHK = 1, 3
      IF (NODCHK.EQ.NODSUB) GO TO 50
      NODE2 = ELEMENT(ELMNBR,NODCHK)
C
C     INCREMENT NODE1'S CONNECTIVITY COUNT
      NUM(NODE1) = NUM(NODE1) + 1
C
C     TEST FOR DUPLICATE NODE NUMBERS IN MAME
      LAST = NUM(NODE1)
      DO 30 COL = 2, LAST
      IF (MAME(NODE1,COL).EQ.NODE2) NUM(NODE1)=NUM(NODE1)-1
      IF (MAME(NODE1,COL).EQ.NODE2) GO TO 50
   30 IF (MAME(NODE1,COL).EQ.0) MAME(NODE1,COL) = NODE2
   50 CONTINUE
C
C     COMPACT THE CONNECTIVITY MATRIX
      ISTART(1)=1
      DO 80 NODE = 1, NONODE
      IF(NODE.NE.NONODE)ISTART(NODE+1)=ISTART(NODE)+NUM(NODE)
      LAST = NUM(NODE)
      DO 70 INDEX = 1, LAST
      IPTR = ISTART(NODE) + INDEX - 1
   70 NAME(IPTR) = MAME(NODE,INDEX)
   80 CONTINUE
C
C     LIST EACH ELEMENT WITH ALL ASSOCIATED NODE NUMBERS
C     THEN PRINT THE CONNECTIVITY MATRIX
      IF (IPRNT(1).EQ.1) CALL OUTPUT(1)
C
      RETURN
      END
```

```
      SUBROUTINE SETIBD
C
C SETIBD - INITIALIZES THE BOUNDARY ARRAY IBD TO ZERO FOR
C NON-BOUNDARY POINTS IN THE GRID AND FIVE FOR THE BOUNDARY
C POINTS. THE ROWS ADJACENT TO THE BOUNDARY ARE SET TO FOUR
C [illegible] NODES NEXT TO THE BOUNDARY WALL.
C THIS [illegible] IS USED DURING EXECUTION FOR TESTING OF
C BOUNDARY POINTS.
C
      INTEGER*2 IBD
      COMMON/CN1/NONODE,NELEMT,NN,HOUR,PHIBAR,NM1,I,NP1,IFLG
      COMMON/CN1A/THETA,THETAB,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/CN5/IBD(156),ICPT,ITRI(156),XI(12),YI(13)
C
C     SET SOUTHREN BOUNDARY POINTS TO 5
      DO 20 NODE = 1, NLONG
   20 IBD(NODE) = 5
C
C     SET SOUTH BOUNDARY + 1 ROW TO 4
      ISTRT = NLONG + 1
      ISTOP = ISTRT + NLONG - 1
      DO 30 NODE = ISTRT, ISTOP
   30 IBD(NODE) = 4
C
C     SET NO-BOUNDARY POINTS TO 0
      ISTRT = ISTOP + 1
      ISTOP = NONODE - 2*NLONG
      DO 40 NODE = ISTRT, ISTOP
   40 IBD(NODE) = 0
C
C     SET NORTH BOUNDARY - 1 ROW TO 4
      ISTRT = ISTOP + 1
      ISTOP = ISTRT + NLONG - 1
      DO 50 NODE = ISTRT, ISTOP
   50 IBD(NODE) = 4
C
C     SET NORTHREN BOUNDARY POINTS TO 5
      ISTRT = ISTOP + 1
      DO 60 NODE = ISTRT, NONODE
   60 IBD(NODE) = 5
C
      RETURN
      END
```

```
      SUBROUTINE CHANAL
C
C     CHANAL - CALCULATES KEY VARIABLES IN THE CHANNEL
C
      COMMON/CHAN/THETA,THETAU,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/[illegible]/THETAO),XL,W,XL1,YTOP,YBOT,XLAMDA,A,[illegible]
      COMMON/CMIX/XMIN,YMIN,[illegible]
C
      PI=3.141593
C
C     OMEGA - EARTH'S ANGULAR VELOCITY
      OMEGA=7.2722052E-05
      XNLAT=NLAT
      XNLONG = NLONG
      RADCON=PI/180.
C
C     THETA - THE CHANNEL'S LOWER LATITUDINAL BOUNDARY
      THETA=THETA*RADCON
C
C     THETAU - THE CHANNEL'S UPPER LATITUDINAL BOUNDARY
      THETAU=THETAU*RADCON
C
C     CONVERT FROM SPHERICAL TO MERCATOR MAP PROJECTIONS
      X=COS(THETAU)/(1.-SIN(THETAU))
      YTOP=ALOG(X)*A
      X=COS(THETA)/(1.-SIN(THETA))
      YBOT=ALOG(X)*A
C
C     W - WIDTH OF THE CHANNEL IN METERS
      W=YTOP-YBOT
C
C     DELTAY - GEOMETRIC SPACE BETWEEN NORTH/SOUTH GRID PTS
      DELTAY=W/XNLAT
C
C     THETAO - CHANNEL'S MID LATITUDE VALUE IN RADIANS
      THETAO=(THETAU-THETA)/2. + THETA
C
C     DELTAX - GEOMETRIC SPACE BETWEEN EAST/WEST GRID PTS
      DELTAX=DELTAY/COS(THETAU)
      XMIN = DELTAX
      YMIN = DELTAY
C
C     XL - THE HORIZONTAL DISTANCE BETWEEN GRID POINTS  (KM)
      XL=XNLONG*DELTAX
C
C     XL1 -  THE HORIZONTAL DISTANCE BETWEEN THE MID
C     CHANNEL GRID POINTS
      XL1=XL*COS(THETAO)
      W1=W*COS(THETAO)
      K=(2.*PI/XL1)**2+(PI/W1)**2
C
C     CORIOLIS VALUE FOR THE MID CHANNEL LATITUDE
      FO = 2.0 * OMEGA * SIN(THETAO)
C
      RETURN
      END
```

```
      SUBROUTINE LOCATE
C
C  LOCATE - CALCULATES THE X, Y GRID POINT VALUES FOR EACH
C  TIME AND STORES IN ARRAYS XLONG AND XLAT. THE VARIABLE
C  XLAMDA IS AN OFFSET VALUE FOR THE (0,0) GRID POINT.
C
C           XLONG - STORES THE HORIZONTAL DISTANCE
C           XLAT  - STORES THE VERTICAL DISTANCES
C
      REAL*8 NUMBER
      COMMON/CM1/MNODE,NDELMT,NN,NHOUR,PHIBAR,NN1,N,NP1,IFLG
      COMMON/CM14/THETA,THETAC,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/CM15/THETA0,XL,H,VEL,YTOP,YBOT,XLAMDA,A,UBAR,F0
      COMMON/CM7/XLONG (155),YLAT(155),AMP,SCOEF,SUM(155)
      COMMON/IPRINT/IPRINT(2)),NAVENO,P1,P2,LEAPFR,PUNISH
C
      PI=3.141593
      RADCON=PI/180.
      FACT = 0.5 * DELTAX
C
C     CALCULATE THE GRID OFFSET VALUE
      XLAMDA=XLAMDA*RADCON
      X0=A*XLAMDA
      LATK=NLAT+1
      KNLONG=NLONG
      K=1
      Y=YBOT
C
      DO 40 I=1,LATK
      XX=X0
C
      DO 30 J=1,NLONG
      XLONG(K)=XX
      XX=XX+DELTAX
      YLAT(K)=Y
      K=K+1
   30 CONTINUE
      Y=Y+DELTAY
      FACT = -FACT
      X0 = X0 - FACT
   40 CONTINUE
C
C     PRINT AREA COORD, AND CARTESIAN COORD
      IF (IPRINT(3).EQ.1) CALL OUTPUT(3)
C
      RETURN
      END
```

```
      SUBROUTINE TRANS
C
C     TRANS - TRANSFORMS THE UNIFORM COORDINATE SYSTEM (X,Y)
C     TO A SPATIALLY VARYING COORDINATE SYSTEM. THIS ROUTINE
C     IS TAKEN FROM ELDER, REF. ?, ONLY THE VARIABLE NAMES
C     HAVE BEEN CHANGED. R1 AND R2 ARE PASSED VIA THE /IPRINT/
C     COMMON BLOCK AND ARE INITIALIZED IN SUBROUTINE INITL.
C
      REAL*4 LX,LY,K1,K2
      REAL*8 [illegible]
      INTEGER*2 HLIMIT,NJM,ISTART,NAME,IHR
      COMMON/CM1/NONODE,NOELMT,NB,HOUR,PHIBAR,IM1,N,HP1,IFLG
      COMMON/CM1A/THETA,THETAJ,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/CM5/THETAO,XL,W,VEL,YTOP,YBOT,XLAMDA,E,UBAR,F)
      COMMON/CM7/XLONG(155),YLAT(155),AMP,SRCOEF,SOR(155)
      COMMON/CM17/XMIN,YMIN,FCSTHR
      COMMON/IPRINT/IPRNT(20),HAVEND,R1,R2,LEAPHR,RUNHR
      DIMENSION SCALEE(11)
C
      IF (R1.EQ.-1.0)) RETURN
      IF ((R1.EQ.1.0).AND.(R2.EQ.1.0))) RETURN
      LX = DELTAX * FLOAT(NLONG)
      LY = DELTAY * FLOAT(NLAT)
      K1 = 6.283185 / LX
      K2 = 6.283185 / LY
      A = (( R1 - 1.0 ) / ( R1 + 1.0 )) / K1
      B = (( R2 - 1.0 ) / ( R2 + 1.0 )) / K2
C
      DO 10 NODE = 1, NONODE
      XX = XLONG(NODE)
      YY = YLAT(NODE)
C     XLONG(NODE) = XX + A * COS(K1*XX) - A
      XLONG(NODE) = XX - A * COS(K1*XX) + A
   10 YLAT(NODE) = YY + B * SIN(K2*YY)
C
      IF (IPRNT(5).EQ.1) CALL OUTPUT(5)
C
      XMIN = 1.0 E+09
      DO 20 NX = 2, NLONG
      DX = XLONG(NX) -XLONG(NX-1)
      IF (XMIN.GT.DX) XMIN = DX
   20 CONTINUE
C
      YMIN = 1.0 E+09
      DO 25 NY = 1, NLAT
      DY = YLAT(NY*NLONG+1) - YLAT(NY*NLONG)
      IF (YMIN.GT.DY) YMIN = DY
   25 CONTINUE
C
C     FOR X OR Y AXIS TRANSFORMATION COMPUTE NEW DT AND
C     ITERATION NUMBER (LEAPHR) FOR A 48 HR FCST
      XYMIN = XMIN
      IF (YMIN.LT.XMIN) XYMIN = YMIN
      DT =(XYMIN/(ABS(UBAR)*(6.0)**.5)))*0.8
      FACTOR = FCSTHR / (DT/3600.)) + 1.0
      LEAPHR = IFIX(FACTOR)
C
C     PLOT X, Y COORDINATES AFTER TRANSFORMATION
      IF (IPRNT(4).EQ.1) CALL OUTPUT(4)
C
      RETURN
      END
```

```
      SUBROUTINE ABRUPT
C
C     ABRUPT - TRANSFORMS THE UNIFORM GRID ESTABLISHED IN LOCATE
C     TO AN ABRUPTLY VARYING GRID MESH. TWO DIFFERENT MESHES ARE
C     AVAILABLE, BOTH ARE STORED IN DATA STATEMENTS.
C
      REAL L1,L2,N1,N2
      REAL M,NU
      INTEGER*2 ELEMNT,NUM,ISTART,NAME,IB)
      COMMON/CM1/NUNODE,NELMT,NM,HOUR,PHSPK,NN1,N,NP1,IFLG
      COMMON/CM14/THETA,THETAD,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/CM16/THETAO,XLAM,VEL,YTOP,YBOT,XLAMDA,E,UBAR,FO
      COMMON/CM17/XLONG(156),YLAT(156),AMP,STCOEF,STM(156)
      COMMON/CM117/XMIN,YMIN,FCSTHR
      COMMON/IPRINT/IPRNT(20),WAVEHG,R1,R2,LEAPHR,RUNTBK
      DIMENSION SCALEE(11)
C     DATA SCALEE/0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,5.5/
      DATA SCALEE/0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,6.0,8.0,10.0/
C
      IF ( 1.NE.-1.0) RETURN
C
C     COMPUTE GRID VALUES FOR ABRUPT GRID PATTERN
C
      NODE = 0
      YMIN = DELTAY
      XMIN = DELTAX
      NLATP1 = NLAT + 1
C
      DO 20 J = 1, NLATP1
      NODE = NODE + 1
      XLONG(NODE) = XLONG(NODE)
      NODE1 = NODE
      DO 10 I = 1, 11
      NODE = NODE + 1
      FACTOR = SCALEE(I)*DELTAX
      XLONG(NODE) = FACTOR + XLONG(NODE1)
      DIFX = XLONG(NODE) - XLONG(NODE-1)
      IF (DIFX.LT.XMIN) XMIN = DIFX
   10 CONTINUE
   20 CONTINUE
C
C     FOR X OR Y AXIS TRANSFORMATION COMPUTE NEW DT AND
C     ITERATION NUMBER (LEAPHR) FOR A 48 HR FCST
      XYMIN = XMIN
      IF (YMIN.LT.XMIN) XYMIN = YMIN
      DT =(XYMIN/(ABS(UBAR)*(6.0**.5)))*0.8
      FACTOR = FCSTHR / (DT/3600.0) + 1.0
      LEAPHR = IFIX(FACTOR)
C
C     PRINT X, Y COORDINATES AFTER TRANSFORMATION
      IF (IPRNT(4).EQ.1) CALL OUTPUT(4)
C
      RETURN
      END
```

```
      SUBROUTINE AREA
C
C     AREA- CALCULATES THE AREA OVER EACH ELEMENT, USING
C     POLYNOMIALS IN X AND Y. DURING THE TRANSFORMATION BETWEEN
C     CARTESIAN (X,Y) AND AREA COORDINATES (L1,L2,L3), THE A
C     AND B COEFFICIENTS FOR EACH NODE PER ELEMENT ARE STORED IN
C     ARRAYS A AND B RESPECTIVELY. FINALLY THE AREA OVER EACH
C     ELEMENT IS CALCULATED BY EVALUATING THE DETERMINATE OF
C     THE COEFFICIENT MATRIX.
C
      INTEGER*2 ELEMNT, ELMNBR
      REAL*8 RUNNBR
      COMMON/CM1/NONODE,NOELMT,NB,HOUR,PHIBAR,NP1,N,NPL,IFLG
      COMMON/CM14/THETA,THETAJ,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/CM2/ELMENT(2,3,3),PHIDIF,ETADIF,DIVDIF
      COMMON/CM7/XLONG(156),YLAT(156),AMP,SRCDEF,SCR(156)
      COMMON/CM8/A(3,288),B(3,288),AREA2(288)
      COMMON/IPRINT/IPRNT(20),NAVEND,K1,K2,LEAPHR,RUNNBR
C
      NODE1 = 1
      NODE2 = 2
      NODE3 = 3
      X = DELTAX * (NLONG + 0)
      ICOUNT = 1
C
C     TRANSFORMATIONS FROM CARTESIAN TO AREA COORDINATES
      DO 10 ELMNBR = 1, NOELMT
      KNODE1 = ELMENT(ELMNBR,NODE1)
      KNODE2 = ELMENT(ELMNBR,NODE2)
      KNODE3 = ELMENT(ELMNBR,NODE3)
      X1 = XLONG(KNODE1)
      X2 = XLONG(KNODE2)
      X3 = XLONG(KNODE3)
C
C     COMPUTE THE A COEFFICIENTS (HORIZONTAL TRANSFORMATION)
      A(NODE1,ELMNBR) = X3 - X2
      A(NODE2,ELMNBR) = X1 - X3
      A(NODE3,ELMNBR) = X2 - X1
C
C     CORRECT END COEFFICIENTS; DUE TO CYCLIC CONTINUITY
      IF (ICOUNT.EQ.(2*NLONG-1)) A(1,ELMNBR)= X3 - X
      IF (ICOUNT.EQ.(2*NLONG-1)) A(3,ELMNBR)= X - X1
      IF (ICOUNT.EQ.(2*NLONG))   A(1,ELMNBR)= X3 - X - X2
      IF (ICOUNT.EQ.(2*NLONG))   A(2,ELMNBR)= X - X3
      IF (ICOUNT.EQ.(4*NLONG-1)) A(1,ELMNBR)= X3 - X
      IF (ICOUNT.EQ.(4*NLONG-1)) A(3,ELMNBR)= X - X1
      IF (ICOUNT.EQ.(4*NLONG))   A(2,ELMNBR)= X1 - X
      IF (ICOUNT.EQ.(4*NLONG))   A(3,ELMNBR)= X + X2 - X1
      IF (ICOUNT.EQ.(4*NLONG)) ICOUNT = 0
C
C     COMPUTE THE B COEFFICIENTS (VERTICAL TRANSFORMATION)
      B(NODE1,ELMNBR) = YLAT(KNODE2) - YLAT(KNODE3)
      B(NODE2,ELMNBR) = YLAT(KNODE3) - YLAT(KNODE1)
      B(NODE3,ELMNBR) = YLAT(KNODE1) - YLAT(KNODE2)
C
C     AREA2 IS 2 TIMES THE AREA OVER THE ELEMENT, THE
C     DETERMINATE OF THE COEFFICIENT MATRIX
      AREA2(ELMNBR) = ABS(A(NODE1,ELMNBR)*B(NODE2,ELMNBR)
     *                - A(NODE2,ELMNBR)*B(NODE1,ELMNBR))
      ICOUNT = ICOUNT + 1
   10 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE INNER
C
C  INNER - CALCULATES ALL THE INNER PRODUCTS NEEDED BY THE
C  MODEL. THESE ARE ALL INDEPENDENT OF TIME SO THEY ARE
C  CALCULATED JUST ONCE DURING THE INITIALIZATION PHASE
C  [illegible]. THE DERIVATION OF THE EQUATIONS
C  [illegible] BUT CRUCIAL TO THE
C  GALERKIN TECHNIQUE AND THE FINITE ELEMENT METHOD. THE
C  EXPLANATION IS MUCH TOO COMPLEX TO INCLUDE HERE. SEE THE
C  MANUSCRIPT FOR DETAILS.
C
      REAL*8 RUNNBR
      COMMON/C12/NOTYPE,NELMT,NN,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/C45/A(3,238),B(3,238),AREA2(238)
      COMMON/C48/VJX(3,238),VJY(3,238),VIJX(3,3,3,238)
      COMMON/C410/VIJ(238),VIJK(3,3,3,238),VIJY(3,3,3,238)
      COMMON/C415/VJXIX(3,3,238),VJYIX(3,3,238)
      COMMON/C415A/VJXIY(3,3,238),VJYIY(3,3,238)
      COMMON/IPRINT/IPRNT(20),NAVEND,R1,R2,LEAPHR,RUNNBR
C
      INDEX = 0
      DO 100 I=1,NELMT
      A4 = AREA2(I)*2
      VIJ(I)=AREA2(I)/24.
      PVIJK = AREA2(I) / 120.0
C
      DO 90 J=1,3
      VJX(J,I)=B(J,I)/6.
      VJY(J,I)=A(J,I)/6.
C
      DO 80 K=1,3
      VJXIX(J,K,I) = B(J,I)*B(K,I)/A4
      VJYIX(J,K,I) = A(J,I)*B(K,I)/A4
      VJXIY(J,K,I) = B(J,I)*A(K,I)/A4
      VJYIY(J,K,I) = A(J,I)*A(K,I)/A4
      KN=1.
      IF (J.EQ.K) KN=2.
      KX=KN/24.
C
      DO 60 M=1,3
      VIJX(J,K,M,I)=B(M,I)*KX
      VIJY(J,K,M,I)=A(M,I)*KX
      X1 = 1.0
      IF ((J.EQ.K).OR.(J.EQ.M).OR.(K.EQ.M))  X1 = 2.0
      IF ((J.EQ.K).AND.(K.EQ.M)) X1 = 6.0
   60 VIJK(J,K,M,I) = PVIJK * X1
   80 CONTINUE
   90 CONTINUE
  100 CONTINUE
C
      IF (IPRNT(5).EQ.1) CALL OUTPUT (5)
C
      RETURN
      END
```

```
      SUBROUTINE AMTRX1
C
C     AMTRX1 - CONSTRUCTS THE COEFFICIENT MATRIX FOR THE L.H.S.
C     OF THE MATRIX EQUATIONS. THESE COEFFICIENTS CONSIST OF
C     THE INNER PRODUCT <VJ,VI>. FOR EACH ELEMENT IN THE DOMAIN
C     THE CONTRIBUTION OF THE TO THE COEFFICIENT MATRIX CAN
C     BE REPRESENTED BY A 3 X 3 MATRIX, BB. THE VALUE OF THE
C     INTEGRATION IS COMPUTED AS FOLLOWS
C
C                    <VI,VJ> = 0              IF | I-J | > 2
C                    <VI,VJ> = AREA2/12       IF   I = J
C                    <VI,VJ> = AREA2/24       IF   I <> J
C
C     THESE COEFFICIENTS STORED IN BB ARE PASSED TO SUBROUTINE
C     ASEMBL WHICH ASSEMBLES THE BB MATRIX INTO THE SHAPE
C     MATRIX, CALLED G.
C
      INTEGER*2 ELMENT
      INTEGER ELMNBR
      COMMON/CM1/NNODE,NOELMT,NN,HOUR,PHIBAR,NTL,N,NP1,IFLG
      COMMON/CM2/ELMENT(288,3),PHIDIF,ZTADIF,DTODIF
      COMMON/CM10/VIJ(288),VIJX(3,3,3,288),VIJY(3,3,3,288)
      COMMON/CM13/H(1044),G(1044),F(1044)
      COMMON/CM14/BB(3,3),C(1044)
C
C     BUILD MASS MATRIX G
      DO 5 I = 1, NN
    5 G(I) = 0.0
      DO 50 ELMNBR = 1, NOELMT
      DO 30 NODEI = 1, 3
      DO 10 NODEJ = 1, 3
      XN=1.
C
C     TEST FOR DIAGONAL MEMBERS
      IF (NODEI.EQ.NODEJ) XN = 2.0
   10 BB(NODEI,NODEJ) = VIJ(ELMNBR)*XN
   30 CONTINUE
      CALL ASEMBL (ELMNBR,G)
   50 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE ASEMBL (ELMNBR,C)
C
C     ASEMBL - CONSTRUCTS A PACKED ONE DIMENSIONAL ARRAY FOR
C     EITHER MASS MATRIX G OR H. THIS ARRAY OR VECTOR HAS THE
C     SAME FORMAT AS ARRAY NAME AND USES BOTH ISTART AND NUM
C     TO LOCATE INFORMATION WITHIN IT. MATRICES G AND H BOTH
C     CONTAIN INNER PRODUCTS (SEE MMATRX FOR DETAILS). THESE
C     INNER PRODUCTS ARE PASSED TO ASEMBL VIA COMMON BLOCK
C     /CM14/ IN MATRIX BB. ASEMBL INSERTS THESE INNER PRODUCTS
C     INTO G OR H FOR ONLY ONE ELEMENT PER CALL, WHERE EACH
C     ELEMENT HAS ITS OWN SET OF INNER PRODUCTS. THE ELEMENT
C     TO BE WORKED ON IS MADE KNOWN TO ASEMBL THROUGH THE INPUT
C     ARGUMENT LIST. ASEMBL ITERATES THROUGH THE CONNECTIVITY
C     LIST FOR EACH NODE BELONGING TO THE ELEMENT CHECKING FOR
C     A MATCH BETWEEN A NODE OF THAT ELEMENT AND A NODE
C     CONTAINED IN ONE OF THE CONNECTIVITY LISTS. ONLY WHEN A
C     MATCH IS FOUND, OR WHICH THERE EXISTS SEVERAL DIFFERENT
C     MATCHES, IS THE INNER PRODUCT ENTERED INTO THE MASS
C     MATRIX. ITS LOCATION DEPENDS ON WHICH NODE OF THE ELEMENT
C     AND WHICH NODE IN THE CONNECTIVITY MATRIX IS MATCHED.
C
C        INPUT ARGUMENTS      ELMNBR - ELEMENT TO BE WORKED ON
C                             C - WORK ARRAY FOR MASS MATRIX
C
      INTEGER*2 ELMENT,NUM,ISTART,NAME
      INTEGER ELMNBR,START
      COMMON/CM2/ELMENT(233,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/CM3/NUM(156),ISTART(156),NAME(1044)
      COMMON/CM14/BB(3,3),CK(1044)
      DIMENSION C(1044)
C
C     LOCATE THE CONNECTIVITY LIST FOR THE NODES OF ELMNBR
      DO 50 INODE = 1, 3
      NODEI = ELMENT(ELMNBR,INODE)
      LAST = NUM(NODEI)
      START = ISTART(NODEI) - 1
C
C     SELECT ONE OF ELMNBR'S NODES, CALL IT NODEJ
      DO 30 JNODE = 1, 3
      NODEJ = ELMENT(ELMNBR,JNODE)
C
C     LOOP THRU NODEI'S CONNECTIVITY, FIND MATCH WITH NODEJ
      DO 10 INDEX = 1, LAST
      NXTNOD = START + INDEX
      TSTNOD = NAME(NXTNOD)
      IF (NODEJ.NE.TSTNOD) GO TO 10
C
C     A MATCH IS FOUND BETWEEN NODEJ AND TSTNOD; ITS INNER
C     PRODUCT STORED IN BB IS ENTERED TO THE MASS MATRIX C
      C(NXTNOD) = BB(INODE,JNODE) + C(NXTNOD)
      GO TO 30
   10 CONTINUE
   30 CONTINUE
   50 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE ARMSET
C
C   ARMSET - TAKEN DIRECTLY FROM OLDER(1981) SETS UP CERTAIN
C   PARAMETERS NEEDED TO INTERPOLATE A VARIABLE RESOLUTION FIELD
C   INTO A UNIFORM FIELD, REQUIRED FOR THE HARMONIC ANALYSIS
C   SCHEME USED IN THIS MODEL. ARMSET CALCULATES THE (X,Y)
C   VALUES FOR EACH POINT TO BE INTERPOLATED. IN ADDITION,
C   IT DETERMINES IN WHICH ELEMENT (NTRI) THIS POINT LIES.
C
      INTEGER*2 ELEMENT,NUM,ISTART,NAME,IB)
      COMMON/C11/NONODE,NELMT,NA,HOUR,PHISNP,IM1,N,NP1,IFLG
      COMMON/C14/THETA,THETAU,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/C12/ELEMNT(256,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/C43/NUM(156),ISTART(156),NAME(1044)
      COMMON/C45/IBD(156),NOPT,NTRI(156),X1(12),Y1(13)
      COMMON/C17/XLONG(156),YLAT(156),NMP,SMCOEF,SOR(156)
      COMMON/C48/A(3,256),B(3,256),AREA2(256)
      DIMENSION XXX(3),YYY(3)
C
      EPSI = .000001
      XL = DELTAX*NLONG
      XM = XL - DELTAX + EPSI
      NYP = NLAT + 1
      NOPT = NONODE
      YS = 0.0
      DO 10 J = 1, NYP
      Y1(J) = YS
   10 YS = YS + DELTAY
      XS = 0.0
      DO 20 I = 1, NLONG
      X1(I) = XS
   20 XS = XS + DELTAX
      I1 = 1
      J1 = 1
      DO 100 I = 1, NOPT
      XX = X1(I1)
      YY = Y1(J1)
      DO 40 J = 1, NELMT
      AREA = 1.001 * AREA2(J)
      DO 30 JK = 1, 3
      X2 = XLONG(ELEMENT(J,JK))
      IF (X2.GT.XM.AND.I1.EQ.1) X2=X2-XL
      IF (X2.LT.-EPSI.AND.I1.EQ.NLONG) X2=X2+XL
      XXX(JK) = X2
   30 YYY(JK) = YLAT(ELEMENT(J,JK))
      XMAX = AMAX1(XXX(1),XXX(2),XXX(3))
      XMIN = AMIN1(XXX(1),XXX(2),XXX(3))
      YMAX = AMAX1(YYY(1),YYY(2),YYY(3))
      YMIN = AMIN1(YYY(1),YYY(2),YYY(3))
      IF (XX.GT.XMAX) GO TO 40
      IF (XX.LT.XMIN) GO TO 40
      IF (YY.GT.YMAX) GO TO 40
      IF (YY.LT.YMIN) GO TO 40
      A1 = ABS(XX*(YYY(3)-YYY(1)) - YY*(XXX(3)-XXX(1))
     1         + XXX(3)*YYY(1) - XXX(1)*YYY(3))
      A2 = ABS(XX*(YYY(2)-YYY(3)) - YY*(XXX(2)-XXX(3))
     1         + XXX(2)*YYY(3) - XXX(3)*YYY(2))
      A3 = ABS(XX*(YYY(1)-YYY(2)) - YY*(XXX(1)-XXX(2))
     1         + XXX(1)*YYY(2) - XXX(2)*YYY(1))
      AA = A1 + A2 + A3
      IF (AA.GT.AREA) GO TO 40
      NTRI(I) = J
      GO TO 50
   40 CONTINUE
   50 I1 = I1 + 1
      IF (I1.GT.NLONG) J1 = J1 + 1
      IF (I1.GT.NLONG) I1 = 1
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE IC
C
C  IC - SETS UP THE INITIAL FIELD ON THE DEPENDENT VARIABLES
C  PHI, PSI AND CHI.
C
      [illegible]
      [illegible] /CMN1/ [illegible]
      [illegible] /CMN1/T[illegible],DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON /CMN4/ [illegible]S(155),AMASSU(155),AMASSI(155),PAST(155,3)
      COMMON /CMN6/ [illegible],XL,W,VEL,YTOP,YBOT,XLAMDA,A,UBAR,FO
      [illegible] /CMN7/XL[illegible](155),YLAT(155),AMP,[illegible]EF,SCR(155)
      COMMON /CMN11/U(156),V(155),XKE(155),PSI(155),CHI(155)
      COMMON /CMN12/[illegible](155,3),DIV(155,3),PHI(155,3),SAVE(155,2)
      COMMON /IPRINT/IPRNT(20)),WAVEND,R1,R2,LEAPHR,RUN[illegible]
C
C  GH - MEAN HEIGHT OF GEOPOTENTIAL HEIGHT
      GH= 9.8 * 5.0E 03
      YK = 3.14159 / W
      XK = 2.0 * 3.14159 /XL
C
C  YMID - HALF THE CHANNEL WIDTH
      YMID = W/2.0 + YBOT
      PLAR = +9[illegible]
      NT=NLAT+1
C
C  COMPUTE THE INITIAL FIELDS FOR U, V AND FI PER MODE
C  AND INITIALIZE ALL THREE WORK VECTORS FOR EACH FIELD
      DO 30 M=1,3
      K=1
      Y = YLAT(1)
      PHIBAR = 0.0
C
C  DO FOR EACH HORIZONTAL (LATITUDINAL) ROW
      DO 20 I=1,NT
      FI1=YK*Y
C
C  DO FOR EACH MODE PER HORIZONTAL ROW
      DO 10 J=1,NLONG
      X=XLONG(K)
      FI2 = XK * X * WAVE[illegible]
      PHI(K,M)=FO*AMP*(SIN(FI1)       )*COS(FI2)-FO*UBAR*(Y-YMID)+GH
      PHIBAR = PHIBAR + PHI(K,M)
      PSI(K) = PHI(K,M)/FO
      BTERM = -((FO*UBAR*XK*(YK**2+XK**2))*AMP)/
     *              ((FO**2)+PLAR*(YK**2+XK**2))
      CTERM = -BTERM/(YK**2+XK**2)
      CHI(K) = CTERM*SIN(FI1)*SIN(FI2)
C     CHI(K) = 0.0
      DIV(K,M) = BTERM*SIN(FI1)*SIN(FI2)
      PAST(K,1) = PHI(K,M)
      PAST(K,2) = PSI(K)
      PAST(K,3) = CHI(K)
      K=K+1
   10 CONTINUE
      IF((I+1).GT.NT) GO TO 20
      Y=YLAT((I+1)*NLONG)
   20 CONTINUE
      PHIBAR = PHIBAR / NOMODE
   30 CONTINUE
      DO 70 I=1,3
      DO 70 J = 1,NOMODE
   70 PHI(J,I) = PHI(J,I)-PHIBAR
C
      IF (IPRNT(7).EQ.1) CALL OUTPUT (7)
C
      RETURN
      END
```

```
      SUBROUTINE DEPVAR
C
C     DEPVAR - INITIALIZES ALL THE DEPENDENT VARIABLES NEEDED
C     TO START THE MODEL AND AFTER EACH TIME INTEGRATION.
C
      COMMON/CMN1/NNODE,NDELT,...,H),...,PHIBAR,NM1,N,NP1,IFLG
      COMMON/CMN4/...S(156),ATNSSJ(156),ATNSSI(156),PAST(156,3)
      COMMON/CMN11/U(156),V(156),KKE(156),PSI(156),CHI(156)
      COMMON/CMN12/ZETA(156,3),DIV(156,3),PHI(156,3),SAVE(156,3)
      COMMON/CMN15/UU(156),UV(156),ALFA(156),BETA(156),UNM1(156)
      COMMON/IPRINT/IPRNT(20),NAVE,...,R1,R2,LEAPHR,RUNHRS
      DIMENSION ZTA(156),DIVRG(156)
C
C     U, HORIZONTAL X COMPONENT OF MOTION
      DO 2 NODE = 1, NNODE
    2 UNM1(NODE) = U(NODE)
      CALL UVRHS (PSI,CHI,2,0)
      CALL SOLVER (U,0,1)
C
C     V, HORIZONTAL Y COMPONENT OF MOTION
      CALL UVRHS (CHI,PSI,2,1)
      CALL SOLVER (V,1,2)
C
C     KE, KINETIC ENERGY (U**2 + V**2)/2
      CALL KERHS
      CALL SOLVER (KKE,0,3)
C
C     ZETA, VORTICITY (DEL**2)*PSI
      CALL VDRHS (PSI,IFLG,1)
      CALL SOLVER (ZTA,1,4)
C
C     DIV, DIVERGENCE (DEL**2)*CHI
      CALL VDRHS (CHI,IFLG,2)
      CALL SOLVER (DIVRG,0,5)
C
C     INITIALIZE NECESSARY TIME LEVELS FOR ZETA AND DIV
      DO 10 NODE = 1, NNODE
      ZETA(NODE,NP1) = ZTA(NODE)
   10 DIV(NODE,NP1) = DIVRG(NODE)
      IF (IFLG.EQ.1) GO TO 30
      DO 20 IL = 1, 3
      DO 20 NODE = 1, NNODE
      UNM1(NODE) = U(NODE)
      ZETA(NODE,IL) = ZTA(NODE)
   20 DIV(NODE,IL)  = DIVRG(NODE)
C
C     UU, U*(ZETA + F)
   30 CALL UUVRHS (U)
      CALL SOLVER (UU,0,6)
C
C     UV  V*(ZETA + F)
      CALL UUVRHS (V)
      CALL SOLVER (UV,1,7)
C
C     ALFA, U*PHI
      CALL ABRHS (U)
      CALL SOLVER (ALFA,0,8)
C
C     BETA, V*PHI
      CALL ABRHS (V)
      CALL SOLVER (BETA,1,9)
C
      IF (IFLG.EQ.1) GO TO 50
      IF (IPRNT(3).EQ.1) CALL OUTPUT(3)
      IF ((IPRNT(15).EQ.1).AND.(IPRNT(19).EQ.1)) CALL PLOTS1
      IFLG = 1
C
   50 RETURN
      END
```

```
      SUBROUTINE UVRHS (D1,D2,NL,ISIGN)
C
C   UVRHS - CALCULATES EITHER THE U OR V COMPONENT OF MOTION
C   DEPENDING ON THE ORDER THE INPUT ARGUMENTS ARE SENT.
C   U = DPSI/DY - DCHI/DX AND V = DCHI/DX + DPSI/DY.
C
C     INPUT ARGUMENTS    D1 - DEPENDENT VAR: PSI FOR U
C                                            CHI FOR V
C                        D2 - DEPENDENT VAR; CHI FOR U
C                                            PSI FOR V
C                        NL - TIME LEVEL FOR PSI AND CHI
C                        ISIGN - SIGN OF THE TERM
C
      INTEGER*2 ELMENT
      INTEGER ELMNBR
      COMMON/CM1/NNODE,NOELMT,NN,NHOUR,PHIBAR,N1,N,NP1,IFLG
      COMMON/CM2/ELMENT(288,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/CM4/RHS(156),AMASSJ(156),AMASSI(156),PAST(156,3)
      COMMON/CM9/VJX(3,288),VJY(3,288),VIJX(3,3,3,288)
      DIMENSION D1(156),D2(156)
C
      CALL RHS0
C
      SIGN = 1.0
      IF (ISIGN.EQ.0) SIGN = -1.0
C
C     CALCULATE FOR EACH NODE PER ELEMENT
      DO 10 ELMNBR = 1, NOELMT
      NODE1 = ELMENT(ELMNBR,1)
      NODE2 = ELMENT(ELMNBR,2)
      NODE3 = ELMENT(ELMNBR,3)
C
C     COMPUTE DERIVATIVE FOR BOTH THE STREAM FUNCTION AND
C     VELOCITY POTENTIAL AND ADD
      TEMP = SIGN * (D1(NODE1)*VJY(1,ELMNBR)
     *              +D1(NODE2)*VJY(2,ELMNBR)
     *              +D1(NODE3)*VJY(3,ELMNBR))
     *            + (D2(NODE1)*VJX(1,ELMNBR)
     *              +D2(NODE2)*VJX(2,ELMNBR)
     *              +D2(NODE3)*VJX(3,ELMNBR))
C
C     ADD RESULTS TO WORK VECTOR RHS
      RHS(NODE1) = RHS(NODE1) + TEMP
      RHS(NODE2) = RHS(NODE2) + TEMP
      RHS(NODE3) = RHS(NODE3) + TEMP
   10 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE RHSO
C
C  RHSO - ZEROING OUT THE ARRAY RHS. THIS ARRAY IS A COMMONLY
C  USED WORK ARRAY, USED FOR RIGHT HAND SIDE TERMS ONLY.
C
      [illegible]
      [illegible] RHS(155),[illegible]SSJ(155),[illegible](155),PAST(155,3)
C
      DO 10 NODE = 1, NONODE
   10 RHS(NODE) = 0.0
C
      RETURN
      END
```

```
      SUBROUTINE SOLVER (Z,IY,ITITLE)
C
C  SOLVER - IS USED TO PERFORM THE RELAXATION FOR SEVERAL
C  VARIABLES. THE FIRST GUESS TO THE SOLUTION IS THE FORCING
C  TERM (RHS) DIVIDED BY THE LUMPED MASS MATRIX. THEN WORK
C  CARRIED OUT THREE ITERATIONS.
C
C     INPUT ARGUMENTS     IY - BOUNDARY TEST FLAG
C                         ITITLE - POINTER FOR CORRECT TITLE
C
C     OUTPUT ARGUMENT     Z - SOLUTION FIELD
C
      INTEGER*2 ELMENT,NUM,ISTART,NAME,IBD
      REAL*8 PUTNB?
      COMMON/CM1/NONODE,NOELMT,NN,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/C15/ NUM(156),ISTART(156),NAME(1044)
      COMMON/C14/RHS(156),AMASSJ(156),AMASSI(156),PAST(156,3)
      COMMON/C45/IBD(156),NOPT,NTRI(156),X1(12),Y1(13)
      COMMON/C115/H(1044),G(1044),F(1044)
      COMMON/IPRINT/IPRNT(20),NAVEND,R1,R2,LEAPER,RUNNER
      DIMENSION Z(156),ONE(156)
      REAL*8 TITLE(8)
      DATA ONE/156*1.0/
      DATA TITLE/'      U  ','      V  ','     KE  ','   ZETA ',
     *'    DIV  ','     DU  ','     DV  ','  ALFA  ','  BETA  '/
C
      ITERAT = 3
      ALPHA = 0.66667
C
C     CALCULATE THE FIRST GUESS
      CALL MASLHS (ONE,G)
      DO 20 NODE = 1, NONODE
      Z(NODE) = 0.0
      IF ((IBD(NODE).EQ.5).AND.(IY.EQ.1)) GO TO 20
      Z(NODE) = RHS(NODE) / (AMASSJ(NODE) + AMASSI(NODE))
   20 CONTINUE
C
      IF (IPRNT(9).EQ.0) GO TO 50
      WRITE(6,30) TITLE(ITITLE)
   30 FORMAT(///,10X,A8,'FIRST GUESS ',/)
      WRITE(6,40) (Z(NODE),NODE=1,NONODE)
   40 FORMAT(1X,12E11.4)
C
C     RELAX FOR 3 ITERATIONS
   50 DO 90 ITER = 1, ITERAT
      CALL MASLHS (Z,G)
      DO 70 NODE = 1, NONODE
      IF ((IBD(NODE).EQ.5).AND.(IY.EQ.1)) GO TO 60
      RESID = (RHS(NODE) -AMASSJ(NODE) - AMASSI(NODE)) /
     *                         G(ISTART(NODE))
      Z(NODE) = Z(NODE) + ALPHA*RESID
      GO TO 70
   60 Z(NODE) = 0.0
   70 CONTINUE
C
      IF (IPRNT(4).EQ.0) GO TO 90
      WRITE(6,80) TITLE(ITITLE), ITER
   80 FORMAT(///,10X,A8,'ITERATION  ',I3,/)
      WRITE(6,40) (Z(NODE),NODE=1,NONODE)
   90 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE MASLHS (DUMY,A)
C
C MASLHS - SETUP FOR MASS MATRIX FOR THE LEFT HAND SIDE.
C MASLHS COMPUTES THE MASS MATRIX IN TWO PARTS. AMASSJ WILL
C CONTAIN THE CONTRIBUTION OF THE SURROUNDING NODES (I<>J)
C AND AMASSI WILL CONTAIN THE CONTRIBUTION FROM ONLY THE
C              I, AS [illegible] BY [illegible].
C
C     INPUT ARGUMENTS   DUMY - DEPENDENT VARIABLE
C                          A - MASS MATRIX
C
      INTEGER*2 NUM,ISTART,NAME
      COMMON/C11/NONODE,NOELMT,NN,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/C13/NUM(156),ISTART(156),NAME(1044)
      COMMON/C14/TAG(156),AMASSJ(156),AMASSI(156),PAST(156,3)
      DIMENSION DUMY(156),A(1044)
C
      ZERO MASS MATRICES
      DO 10 NODE = 1, NONODE
      AMASSJ(NODE) = 0.0
   10 AMASSI(NODE) = 0.0
C
      SUM THE I<>J CONTRIBUTIONS
      DO 100 NODE = 1, NONODE
      SUMJ = 0.0
      ISTRT = ISTART(NODE) + 1
      LAST = ISTRT + NUM(NODE) - 2
      DO 50 INDEX = ISTRT, LAST
      JNODE = NAME(INDEX)
   50 SUMJ = SUMJ + DUMY(JNODE)*A(INDEX)
C
      STORE THE I=J CONTRIBUTIONS
      AMASSI(NODE) = AMASSI(NODE)+DUMY(NODE)*A(ISTART(NODE))
      AMASSJ(NODE) = AMASSJ(NODE)+SUMJ
  100 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE KERHS
C
C KERHS - CALCULATES THE KINETIC ENERGY CONTRIBUTION, WHERE
C KE = (U**2 + V**2)/2.
C
      [illegible],ISTART,NAME
      [illegible]
      [illegible](156),ISTART(156),NAME(1044)
      [illegible]RHS(156),[illegible](156),[illegible](156),PAST(156,3)
      COMMON /CBL11/ U(156),V(156),AREA(156),PSI(156),CHI(156)
      [illegible](3,3),C(1044)
      DIMENSION DUMY(156,3)
C
      CALL RHS0
C
C     COMPUTE U**2
      DO 10 I = 1, NM
   10 C(I) = 0.0
      CALL TERM1 (1,U,DUMY,1,0)
C
      DO 20 NODE = 1, NONODE
      IFIRST = ISTART(NODE)
      LAST = IFIRST + NUM(NODE) - 1
      DO 20 INDEX = IFIRST, LAST
      RHS(NODE) = RHS(NODE) + U(NAME(INDEX))*C(INDEX)
   20 CONTINUE
C
C     COMPUTE V**2
      DO 30 I = 1,NM
   30 C(I) = 0.0
      CALL TERM1 (2,V,DUMY,1,0)
C
      DO 70 NODE = 1, NONODE
      IFIRST = ISTART(NODE)
      LAST = IFIRST + NUM(NODE) - 1
      DO 50 INDEX = IFIRST, LAST
      RHS(NODE) = RHS(NODE) + V(NAME(INDEX))*C(INDEX)
   50 CONTINUE
   70 CONTINUE
C
      DO 100 NODE = 1, NONODE
  100 RHS(NODE) = 0.5*RHS(NODE)
C
      RETURN
      END
```

```
      SUBROUTINE VORHS (DL,IFLG,IPTR)
C
C     VORHS - CALCULATES THE VORTICITY OR DIVERGENCE DEPENDING
C     ON THE VALUE OF THE INPUT ARGUMENTS. THE CALCULATION
C     IS VORTICITY = (DEL**2)*PSI, DIVERGENCE = (DEL**2)*CHI
C
C     INPUT ARGUMENTS      DL - DEPENDENT VAR, PSI OR CHI
C                          IFLG - FLAG FOR FIRST TIME STEP
C                          IPTR - POINTER FOR FIELD
C
      COMMON/CM12/ZETA(156,3),DIV(156,3),PHI(156,3),SAVE(156,2)
      COMMON/CM15//VJXIX(3,3,233),VJYIX(3,3,233)
      COMMON/CM16/VJXIY(3,3,233),VJYIY(3,3,233)
      COMMON/CM13/DU(156),DV(156),ALFA(156),BETA(156),UNIT(156)
      DIMENSION DL(156),DUMY(156,3)
C
      CALL RHSO
C
C     COMPUTE VALUE OF FORCING TERMS FOR FIRST TIME STEP
      IF (IFLG.NE.0) GO TO 10
C     -(PSI OR CHI)*<VJX,VIX>
      SIGN = -1.0
      CALL TERM2 (DL,DUMY,1,NL,VJXIX,SIGN,1)
C
C     -(PSI OR CHI)*<VJY,VIY>
      CALL TERM2 (DL,DUMY,1,NL,VJYIY,SIGN,2)
      GO TO 80
C
C     COMPUTE VALUE OF FORCING TERMS FOR VORTICITY
   10 IF (IPTR.NE.2) GO TO 50
      DO 20 NODE = 1, NONODE
      RHS(NODE) = SAVE(NODE,1)
      IF (ISO(NODE).EQ.5) RHS(NODE) = 0.0
   20 CONTINUE
      IF (IFLG.NE.1) GO TO 30
      CALL SOLVER (ZTA,1,4)
   30 CONTINUE
      GO TO 80
C
C     COMPUTE VALUE OF FORCING TERM FOR DIVERGENCE
   50 DO 60 NODE = 1, NONODE
      RHS(NODE) = SAVE(NODE,2)
      IF (ISO(NODE).EQ.5) RHS(NODE) = 0.0
   60 CONTINUE
      IF (IFLG.NE.1) GO TO 70
      CALL SOLVER (DIVRG,0,5)
   70 CONTINUE
   80 RETURN
      END
```

```
      SUBROUTINE UVRHS (DUMY)
C
C     UVRHS - CALCULATES THE UU OR VV TERMS RHS VALUE, WHERE
C     UU = U*(ZETA + F) AND VV = V*(ZETA + F).
C
C        INPUT ARGUMENT    DUMY - DEPENDENT VARIABLE U OR V
C
      INTEGER R2, NUM, ISTART, NAME
      COMMON/CM1/NONODE,NOELMT,NN,IOUR,PHIBAR,ITL,M,MP1,IFLG
      COMMON/CM3/NUM(156),ISTART(156),NAME(1044)
      COMMON/CM4/RHS(156),AMASSU(156),AMASSI(156),PAST(156,3)
      COMMON/CM5/THETAD,KL,N,VEL,YTOP,YBOT,XLAMDA,E,UBAR,F0
      COMMON/CM12/ZETA(156,3),DIV(156,3),PHI(156,3),SAVE(156,2)
      COMMON/CM13/A(1044),G(1044),F(1044)
      COMMON/CM14/BB(3,3),C(1044)
      DIMENSION DUMY(156),TEMP(156)
C
      CALL RHS0
C
C     CONSTRUCT (U OR V)*ZETA TERM
      DO 10 I = 1, NN
   10 C(I) = 0.0
      CALL TERM1 (1,DUMY,ZETA,2,MP1)
C
      DO 20 NODE = 1, NONODE
      IFIRST = ISTART(NODE)
      LAST = IFIRST + NUM(NODE) - 1
      DO 20 INDEX = IFIRST, LAST
      RHS(NODE) = RHS(NODE) + DUMY(NAME(INDEX))*C(INDEX)
   20 CONTINUE
C
C     CONSTRUCT (U OR V)*F0 TERM
      DO 30 NODE = 1, NONODE
   30 TEMP(NODE) = 0.0
      DO 40 NODE = 1, NONODE
      IFIRST = ISTART(NODE)
      LAST = IFIRST + NUM(NODE) - 1
      DO 40 INDEX = IFIRST, LAST
      TEMP(NODE) = TEMP(NODE)+DUMY(NAME(INDEX))*G(INDEX)
   40 CONTINUE
      DO 50 NODE = 1, NONODE
   50 RHS(NODE) = RHS(NODE) + F0*TEMP(NODE)
C
      RETURN
      END
```

```
      SUBROUTINE XBRHS (DUMY)
C
C  XBRHS - CALCULATES THE RHS TERM FOR THE ALFA OR BETA
C  FIELDS, DEPENDING ON WHETHER THE U OR V COMPONENT OF
C  MOTION IS MULTIPLIED WITH THE HEIGHT FIELD.
C
C     INPUT ARGUMENTS: DUMY - DEPENDENT VARIABLE (U OR V)
C
      INTEGER*2 ELMENT,NUM,ISTART,NAME,IB
      COMMON/CM1/NNODE,NELMT,NT,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/CM3/NUM(156),ISTART(156),NAME(1044)
      COMMON/CM4/RHS(156),AMASSJ(156),AMASSI(156),PAST(156,3)
      COMMON/CM12/ZETA(156,3),DIV(156,3),PHI(156,3),SAVE(156,2)
      COMMON/CM14/B(3,3),C(1044)
      DIMENSION DUMY(156),FILLER(156)
C
      CALL RHSD
C
C     COMPUTE (U OR V)*PHI
      DO 10 I = 1, M
   10 C(I) = 0.0
      CALL TERM1 (1,FILLER,PHI,2,NP1)
C
      DO 20 NODE = 1, NNODE
      IFIRST = ISTART(NODE)
      LAST = IFIRST + NUM(NODE) - 1
      DO 20 INDEX = IFIRST, LAST
      RHS(NODE) = RHS(NODE) + DUMY(NAME(INDEX))*C(INDEX)
   20 CONTINUE
C
      RETURN
      END
```

D. FORECAST PHASE

```
      SUBROUTINE NTFIND (DT)
C
C  NTFIND - IS THE ROUTINE THAT IS USED TO COMPUTE THE
C  FIRST FORWARD TIME INTEGRATION. THIS IS DONE BY TWO HALF
C  FORWARD STEPS; F(N+1) = F(N) + DT/2*K.
C
C     INPUT ARGUMENT     DT - DELTA T TIME INTERVAL
C
      REAL*8 RUNNBR
      COMMON/CM1/NONODE,NOELMT,NW,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/CM11/H(156),Z(156),KNE(156),PSI(156),CHI(156)
      COMMON/CM12/ZETA(156,3),DIV(156,3),PHI(156,3),SAVE(156,2)
      COMMON/CM13/A(1044),B(1044),C(1044)
      COMMON/IPRINT/IPRNT(20),WAVENO,X1,X2,LEAPFR,RUNNBR
      DIMENSION PHINEW(156)
C
C     INITIALIZE THE TIME LEVELS
      NM1 = 1
      N = 1
      NP1 = 3
      IHOUR = 0
C
      DTHALF = DT/2.0
C
C     CONSTRUCT MASS MATRIX NEEDED FOR SOLUTION OF CONTEQ
      CALL MATRX2 (DTHALF)
C
C     PERFORM TWO HALF STEPS
      DO 20 ISTEP = 1, 2
C
C     UPDATE FCST TIME
      HOUR = HOUR + DTHALF/3600.0
C
      CALL VORTEQ (DTHALF)
      CALL RELAX (PSI,H,2)
C
      CALL CONTEQ (DTHALF)
      CALL RELAX1 (PHINEW,F,1)
      DO 10 NODE = 1, NONODE
   10 PHI(NODE,NP1) = PHINEW(NODE)
C
      CALL DIVEQ  (DTHALF,1)
      CALL RELAX (CHI,H,3)
C
      CALL DEPVAR
C
C     TEST FOR OUTPUT CONDITIONS
      IHOUR = IHOUR + 3
      IF ((IPRNT(12)*2).NE.IHOUR) GO TO 15
      IF (IPRNT(11).EQ.1) CALL OUTPUT (7)
      IF (IPRNT(13).EQ.1) CALL OUTPUT (5)
      IF (IPRNT(14).EQ.1) CALL OUTPUT (8)
      IF (IPRNT(15).EQ.1) CALL PLOTS1
      IHOUR = 0
C
C     UPDATE THE TIME LEVEL POINTERS
   15 NM1 = 3
      N = 3
   20 NP1 = 2
C
      RETURN
      END
```

```
      SUBROUTINE MATRX2 (DT)
C
C     MATRX2 - CONSTRUCTS THE COEFFICIENT MATRIX FOR THE L.H.S.
C     OF THE MATRIX EQUATIONS USED IN THE SOLUTION OF THE THREE
C     GOVERNING EQUATIONS. TWO MATRICES ARE CONSTRUCTED: H
C     WHICH IS USED TO SOLVE VORTICITY AND DIVERGENCE AND F, USED
C     TO SOLVE THE CONTINUITY EQUATION. THEY ARE ASSEMBLED IN THE
C     SAME MANNER THAT THE G MATRIX WAS IN MATRX1. THE DIFFERENCE
C     LIES IN THE INNER PRODUCTS USED. H IS CONSTRUCTED WITH
C     THE SUM OF <VJX,VIX> + <VJY,VIY> INNER PRODUCTS. F IS
C     CONSTRUCTED BY SUMMING THE H AND G MATRICES, WHERE G IS
C     MULTIPLIED BY A CONSTANT WHICH CONTAINS THE CURRENT DELTA
C     T FOR THE TIME INTEGRATION. F THEREFORE MUST BE RECOMPUTED
C     WHENEVER THE DELTA T CHANGES AND IT CHANGES TWICE. HALF
C     DELTA T IS USED FOR THE MATSUNO TECHNIQUE AND TWICE THE
C     DELTA T IS USED IN THE LEAPFROG TECHNIQUE.
C
C     INPUT ARGUMENT        DT - CURRENT DELTA T
C
      INTEGER*2 ELMENT
      REAL*8 RUNNBR
      INTEGER ELMNBR
      COMMON/CM1/NONODE,NOELMT,NN,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/CM2/ELMENT(283,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/CM13/H(1044),G(1044),F(1044)
      COMMON/CM14/BB(3,3),C(1044)
      COMMON/CM15/VJXIX(3,3,283),VJYIX(3,3,283)
      COMMON/CM15A/VJXIY(3,3,283),VJYIY(3,3,283)
      COMMON/IPRINT/IPRNT(20),HAVEND,K1,K2,LEAPHR,RUNNBR
C
      DO 10 INDEX = 1, NN
   10 H(INDEX) = 0.0
C
C     BUILD MASS MATRIX H, WITH INNER PRODUCT <VJX,VIX>
      DO 60 ELMNBR = 1, NOELMT
      DO 30 NODEI = 1, 3
      DO 20 NODEJ = 1, 3
   20 BB(NODEI,NODEJ) = VJXIX(NODEI,NODEJ,ELMNBR)
   30 CONTINUE
      CALL ASEMBL (ELMNBR,H)
      DO 50 NODEI = 1, 3
      DO 40 NODEJ = 1, 3
   40 BB(NODEI,NODEJ) = VJYIY(NODEI,NODEJ,ELMNBR)
   50 CONTINUE
      CALL ASEMBL (ELMNBR,H)
   60 CONTINUE
C
C     BUILD MASS MATRIX FOR CONTED
      CONST = 4.0/(PHIBAR*(DT**2))
      DO 80 INDEX = 1, NN
   80 F(INDEX) = H(INDEX) + CONST*G(INDEX)
C
      IF (IPRNT(2).EQ.1) CALL OUTPUT(2)
C
      RETURN
      END
```

```
      SUBROUTINE VORTEQ (DT)
C
C  VORTEQ - CALCULATES THE RHS CONTRIBUTION (FORCING TERM)
C  OF THE VORTICITY PROGNOSTIC EQUATION. EACH TERM'S
C  CONTRIBUTION IS COMPUTED AND STORED IN ARRAY RHS.
C
C     INPUT ARGUMENTS:   DT - DELTAT VALUE
C
      INTEGER*2 ELEMNT
      COMMON/CM1/ [illegible],PI,IFLG
      COMMON/CM2/ ELEMNT(233,3),PHIDIF,ZTADIF,VORDIF
      COMMON/CM4/ RHS(156),AMASSJ(156),AMASSI(156),PAST(156,3)
      COMMON/CM9/ VJX(3,233),VJY(3,233),VIJK(3,3,3,233)
      COMMON/CM12/ZETA(156,3),DIV(156,3),PHI(156,3),SAVE(156,2)
      COMMON/CM15/VJXIX(3,3,233),VJYIX(3,3,233)
      COMMON/CM15A/VJXIY(3,3,233),VJYIY(3,3,233)
      COMMON/CM16/QU(156),QV(156),ALPHA(156),BETA(156),[illegible](156)
      DIMENSION DUMMY(156)
C
      CALL RHS0
C
C     -DT*<QUJ.VJX,VI>
      SIGN = -DT
      CALL TERM4 (QU,VJX,SIGN,1)
C
C     -DT*<QVJ.VJY,VI>
      CALL TERM4 (QV,VJY,SIGN,2)
C
C     +<ZETA(N-1)J.VJ,VI>
      SIGN = 1.0
      CALL TERM3 (DUMMY,ZETA,2,NM1,SIGN)
C
C     -ZTADIF*DT(<ZETA.VJX,VIX>+<ZETA.VJY,VIY>)
      SIGN =  -ZTADIF*DT
      CALL TERM2 (DUMMY,ZETA,2,NM1,VJXIX,SIGN,1)
      CALL TERM2 (DUMMY,ZETA,2,NM1,VJYIY,SIGN,2)
C
      DO 10 NODE = 1, NONODE
   10 SAVE(NODE,1) = RHS(NODE)
C
      RETURN
      END
```

```
      SUBROUTINE RELAX (Z,A,IPTR)
C
C     RELAX - IS CALLED FOR RELAXATION OF ALL PROGNOSTIC
C     EQUATIONS.
C
C        INPUT ARGUMENT        Z - CONTAINS THE SOLVED FIELD
C                              A - MASS MATRIX
C                              IPTR - POINTER TO THE FIELD DATA
C
      INTEGER*2 NUM,ISTART,NAME
      REAL*8 FIELD
      COMMON/C11/NONODE,NDELAT,NT,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/C11A/THETA,THETAO,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/C13/NUM(156),ISTART(156),NAME(1044)
      COMMON/C14/RHS(156),XMASSJ(156),XMASSI(156),PAST(156,3)
      DIMENSION Z(156),EPSLN(3),ALF(3),A(1044),FIELD(3)
      DATA MAXITR/200/,FIELD/' PHI     ',' PSI     ',' CHI     '/
      DATA EPSLN/1.0E-01,5.0E+02,1.5E+01/,ALF/1.30,1.29727,1.582/
C
      IF (IPTR.EQ.2) WRITE(6,5) HOUR
    5 FORMAT(/,10X,'--------------',F4.1,' HRS     ------------')
      DO 10 NODE = 1, NONODE
      IF (IPTR.EQ.1) RHS(NODE) = -RHS(NODE)
   10 Z(NODE) = PAST(NODE,IPTR)
      EPSILN = EPSLN(IPTR)
      ALPHA = ALF(IPTR)
      IFIRST = NLONG + 1
      LAST = NLONG*NLAT
      ITER = 0
      IDIR = 1
C
C     OVER RELAX TO SOLVE SOLUTION
   20 ITER = ITER + 1
      NODE = IFIRST
      IF (IDIR.EQ.-1) NODE = LAST
      ITOTAL = 0
C
C     COMPUTE J AND I CONTRIBUTIONS
      DO 40 INODE = IFIRST, LAST
      XMJ = 0.0
      ISTRT = ISTART(NODE) + 1
      ILAST = ISTRT + NUM(NODE) - 2
      DO 30 JJ = ISTRT , ILAST
      JNODE = NAME(JJ)
   30 XMJ = XMJ + Z(JNODE)*A(JJ)
      XMI = Z(NODE)*A(ISTART(NODE))
C
C     COMPUTE RESIDUAL
      RESID = ( RHS(NODE) + XMJ + XMI)/A(ISTART(NODE))
      IF(ABS(RESID).LT.EPSILN) ITOTAL = ITOTAL + 1
      Z(NODE) = Z(NODE) - ALPHA*RESID
      NODE = NODE + IDIR
   40 CONTINUE
      IDIR = -IDIR
C
C     SET BOUNDARY CONDITIONS
      CALL BDRY (Z,IPTR)
C
C     TEST FOR COMPLETION
      IF (ITOTAL.EQ.(NONODE-2*NLONG)) GO TO 50
      IF (ITER.NE.MAXITR) GO TO 20
C
   50 DO 60 NODE = 1, NONODE
   60 PAST(NODE,IPTR) = Z(NODE)
      WRITE(6,70) ITER,FIELD(IPTR),ALPHA,EPSILN
   70 FORMAT(10X,I3,' ITERATIONS ',A8,
     *10X,'ALPHA = ',F7.4,'      EPSILON = ',F15.6)
C
      RETURN
      END
```

```
      SUBROUTINE BDRY (Z,IPTR)
C
C BDRY - APPLIES THE BOUNDARY CONDITIONS TO THE PHI, PSI
C AND CHI FIELDS DURING THEIR RELAXATION.
C
C     INPUT ARGUMENT        Z - FIELD BEING RELAXED
C                           IPTR - POINTER TO FIELD DATA
C                                  1 = PHI
C                                  2 = PSI
C                                  3 = CHI
C
      COMMON/C11/NODE,NHELMI,NN,HOUR,PHIBAR,NAL,N,NP1,IFLD
      COMMON/C11A/THETA,THETAU,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/C15/THETAM,XL,H,VEL,YTOP,YBOT,XLAMDA,E,UBAR,F0
      COMMON/C17/XLONG(150),YLAT(150),AMP,SCOFF,SQF(150)
      COMMON/C11/U(150),V(150),XNE(150),PSI(150),CHI(150)
      DIMENSION Z(150)
      DATA PSIN/0.1204E+10/,PSIS/0.1519E+10/
C
      LAST = NLONG*NLAT
      GO TO (10,30,50), IPTR
C
C     SET BOUNDARY CONDITIONS FOR PHI FIELD
   10 CONTINUE
      GO TO 70
C
C     SET BOUNDARY CONDITIONS FOR PSI FIELD
   30 DO 40 NODE = 1, NLONG
      Z(NODE)= PSIS
   40 Z(LAST + NODE) = PSIN
      GO TO 70
C
C     SET BOUNDARY CONDITIONS FOR CHI FIELD
   50 DO 60 NODE = 1, NLONG
      Z(NODE) = 0.0
      NODE2 = NODE + LAST
   60 Z(NODE2) = 0.0
C
   70 RETURN
      END
```

```
      SUBROUTINE CONTFO (DT)
C
C  CONTFO - CALCULATES THE RHS CONTRIBUTION (FORCING TERM)
C  FOR THE CONTINUITY PROGNOSTIC EQUATION.
C
C     INPUT ARGUMENT      DT - DELTA T VALUE
C
      INTEGER*2 ELEMNT
      COMMON/CM1/NODE,NELMNT,NN,NUJK,PHIBAR,NN1,N,NP1,IHLB
      COMMON/CM12/ELEMNT(233,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/CM7/XLONG(156),YLAT(156),AVP,SMCOFF,STP(156)
      COMMON/CM9/VJX(3,233),VJY(3,233),VIJX(3,3,3,233)
      COMMON/CM11/Q(156),V(156),XKE(156),PSI(156),CHI(156)
      COMMON/CM12/ZETA(156,3),DIV(156,3),PHI(156,3),SAVE(156,2)
      COMMON/CM15/VJXIX(3,3,233),VJYIX(3,3,233)
      COMMON/CM15A/VJXIY(3,3,233),VJYIY(3,3,233)
      COMMON/CM16/QU(156),QV(156),ALFA(156),BETA(156),[illegible](156)
      DIMENSION DUMMY(156),DUMMY2(156,3),XCHI(156),DMP1(156)
      DATA XCHI/156*0.0/
C
      CALL JVRHS (PSI,XCHI,2,0)
      CALL SOLVER (DMP1,0,1)
C
      CALL RHSO
C
C     COMPUTE THE BOUNDARY CONTRIBUTION
      SIGN = 1.0
      CALL PHIBDY (DMP1,SIGN)
      CALL PHIBDY (DUMMY,SIGN)
C
C     +CONST*<PHI(N-1)J.VJ,VI>
      SIGN = 4.0/(PHIBAR*(DT**2))
      CALL TERM3 (DUMMY,PHI,2,NN1,SIGN)
C
C     -4.0/DT*<DIV(N-1)J.VJ,VI>
      SIGN = -4.0/DT
      CALL TERM3 (DUMMY,DIV,2,NN1,SIGN)
C
C     -1.0*(<PHI(N-1)J.VJX,VIX>+<PHI(N-1)J.VJY,VIY>)
      SIGN = -1.0
      CALL TERM2 (DUMMY,PHI,2,NN1,VJXIX,SIGN,1)
      CALL TERM2 (DUMMY,PHI,2,NN1,VJYIY,SIGN,1)
C
C     -2.0*<QVJ.VJX,VI>
      SIGN = -2.0
      CALL TERM4 (QV,VJX,SIGN,1)
C
C     +2.0*<QUJ.VJY,VI>
      SIGN = 2.0
      CALL TERM4 (QU,VJY,SIGN,1)
C
C     -2.0*(<XKEJ.VJX,VIX>+<XKEJ.VJY,VIY>)
      SIGN = -2.0
      CALL TERM2 (XKE,DUMMY2,1,1,VJXIX,SIGN,1)
      CALL TERM2 (XKE,DUMMY2,1,1,VJYIY,SIGN,1)
C
C     +2.0*DIVDIF(<DIV.VJX,VIX>+<DIV.VJY,VIY>)
      SIGN = 2.0*DIVDIF
      CALL TERM2 (DUMMY,DIV,2,NN1,VJXIX,SIGN,1)
      CALL TERM2 (DUMMY,DIV,2,NN1,VJYIY,SIGN,2)
C
C     -4.0/(PHIBAR*DT)*<ALFAJ.VJX,VI>
      SIGN = -4.0/(PHIBAR*DT)
      CALL TERM4 (ALFA,VJX,SIGN,1)
C
C     -4.0/(PHIBAR*DT)*<BETA.VJY,VI>
      SIGN = -4.0/(PHIBAR*DT)
      CALL TERM4 (BETA,VJY,SIGN,1)
      RETURN
      END
```

```
      SUBROUTINE RHSDY (UU,SIGN)
C
      COMMON/CM1/NNODE,NDELAT,NA,[illegible],PHIBAR,NML,N,NP1,IFLG
      COMMON/CM1A/THETA,THETAU,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/CM2/RHS(156),MASSU(156),[illegible](156),PAST(156,3)
      COMMON/CM3/[illegible]
      COMMON/CM4/[illegible](156),XLAT(156),[illegible],SRCNODE,S[illegible](156)
      COMMON/[illegible]/IPR[illegible](20)),[illegible],L1,K2,[illegible]
      DIMENSION UU(156)
C
      LAST = NLONG * NLAT
      LONGM1 = NLONG - 1
      SIGN = SIGN
C
      DO 10 NODE = 2, LONGM1
      NODES = NODE
      NODEN = LAST + NODE
      DX1 = XLONG(NODES) - XLONG(NODES-1)
      DX2 = XLONG(NODES+1) - XLONG(NODES)
      RHS(NODES) =  SIGN*F0*(DX1*UU(NODES-1)
     1            +  2.0*DX1*UU(NODES) + 2.0*DX2*UU(NODES)
     2            +  DX2*UU(NODES+1))/6.0
      DX1 = XLONG(NODEN) - XLONG(NODEN-1)
      DX2 = XLONG(NODEN+1) - XLONG(NODEN)
   10 RHS(NODEN) = -SIGN*F0*(DX1*UU(NODEN-1)
     1            +  2.0*DX1*UU(NODEN) + 2.0*DX2*UU(NODEN)
     2            +  DX2*UU(NODEN+1))/6.0
C
C     CYCLIC CONTINUITY FOR EAST BOUNDARY POINTS
      DX1 = XL - XLONG(NLONG)
      DX2 = XLONG(2) - XLONG(1)
      RHS(1)      =  SIGN*F0*(DX1*UU(NLONG)
     1            +  2.0*DX1*UU(1) + 2.0*DX2*UU(1)
     2            +  DX2*UU(2))/6.0
      DX1 = XL - XLONG(NONODE)
      DX2 = XLONG(LAST+2) - XLONG(LAST+1)
      RHS(LAST+1)= -SIGN*F0*(DX1*UU(NODEN-1)
     1            +  2.0*DX1*UU(NODEN) + 2.0*DX2*UU(NODEN)
     2            +  DX2*UU(NODEN+1))/6.0
C
C     CYCLIC CONTINUITY FOR WEST BOUNDARY POINTS
      DX1 = XLONG(NLONG) - XLONG(NLONG-1)
      DX2 = XL - XLONG(NLONG)
      RHS(NLONG) =  SIGN*F0*(DX1*UU(NLONG-1)
     1            +  2.0*DX1*UU(NLONG) + 2.0*DX2*UU(NLONG)
     2            +  DX2*UU(1))/6.0
      DX1 = XLONG(NONODE) - XLONG(NONODE-1)
      DX2 = XL - XLONG(NONODE)
      RHS(LAST+NLONG) = -SIGN*F0*(DX1*UU(NONODE-1)
     1            +  2.0*DX1*UU(NONODE) + 2.0*DX2*UU(NONODE)
     2            +  DX2*UU(LAST+1))/6.0
C
      RETURN
      END
```

```
      SUBROUTINE RELAX1 (Z,A,IPTR)
C
C     RELAX1 - IS CALLED FOR RELAXATION OF THE PHI PROGNOSTIC
C     EQUATION.
C
C     INPUT ARGUMENT          Z - CONTAINS THE SOLVED FIELD
C                             A - MASS MATRIX
C                             IPTR - POINTER TO THE FIELD DATA
C
      INTEGER*2 NUM,ISTART,NAME
      REAL*8 FIELD
      COMMON/CM1/NONODE,NDELMT,NN,NBOUN,PHIBAR,NM1,N,NP1,IFLG
      COMMON/CM1A/THETA,THETAS,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/CM3/NUM(156),ISTART(156),NAME(1044)
      COMMON/CM4/RHS(156),AMASSJ(156),AMASSI(156),PAST(156,3)
      DIMENSION Z(156),EPSLN(3),ALF(3),A(1044),FIELD(3)
      DATA MAXITR/200/,FIELD/' PHI    ',' PSI    ',' CHI    '/
      DATA EPSLN/1.0E-01,5.0E+02,1.0E+03/,ALF/1.40,1.20,1.520/
C
      DO 10 NODE = 1, NONODE
      IF (IPTR.EQ.1) RHS(NODE) = -RHS(NODE)
   10 Z(NODE) = PAST(NODE,IPTR)
      EPSILN = EPSLN(IPTR)
      ALPHA = ALF(IPTR)
      IFIRST = 1
      LAST = NLONG*NLAT + NLONG
C     IFIRST = NLONG + 1
C     LAST = NLONG*NLAT
      ITER = 0
      IDIR = 1
C
C     OVER RELAX TO SOLVE SOLUTION
   20 ITER = ITER + 1
      NODE = IFIRST
      IF (IDIR.EQ.-1) NODE = LAST
      ITOTAL = 0
C
C     COMPUTE J AND I CONTRIBUTIONS
      DO 40 INODE = IFIRST, LAST
      XMJ = 0.0
      ISTRT = ISTART(NODE) + 1
      ILAST = ISTRT + NUM(NODE) - 2
      DO 30 JJ = ISTRT , ILAST
      JNODE = NAME(JJ)
   30 XMJ = XMJ + Z(JNODE)*A(JJ)
      XMI = Z(NODE)*A(ISTART(NODE))
C
C     COMPUTE RESIDUAL
      RESID = ( RHS(NODE) + XMJ + XMI)/A(ISTART(NODE))
      IF(ABS(RESID).LT.EPSILN) ITOTAL = ITOTAL + 1
      Z(NODE) = Z(NODE) - ALPHA*RESID
      NODE = NODE + IDIR
   40 CONTINUE
      IDIR = -IDIR
C
C     TEST FOR COMPLETION
      IF (ITOTAL.EQ.NONODE) GO TO 50
      IF (ITER.NE.MAXITR) GO TO 20
C
   50 DO 60 NODE = 1, NONODE
   60 PAST(NODE,IPTR) = Z(NODE)
      WRITE(6,70) ITER,FIELD(IPTR),ALPHA,EPSILN
   70 FORMAT(10X,I3,' ITERATIONS ',A8,
     *10X,'ALPHA = ',F7.4,'      EPSILON = ',F15.6)
C
      RETURN
      END
```

```
      SUBROUTINE DIVRO (DT,IMATZU)
C
C  DIVRO - CALCULATES THE RHS CONTRIBUTION (FORCING TERM)
C  FOR THE DIVERGENCE PROGNOSTIC EQUATION.
C
C     INPUT ARGUMENT      DT - DELTAT VALUE
C                         IMATZU - FLAG FOR FIRST 2 TIME STEPS
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/CM1/NONODE,NELMT,NN,NOJN,PHIBAR,NM1,N,NP1,IFLG
      COMMON/CM2/ELMENT(233,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/CM4/RHS(156),AMASSJ(156),AMASSI(156),PAST(156,3)
      COMMON/CM9/VJX(3,233),VJY(3,233),VIJK(3,3,3,233)
      COMMON/CM11/U(156),V(156),AKE(156),PSI(156),CHI(156)
      COMMON/CM12/ZETA(156,3),DIV(156,3),PHI(156,3),SAVE(156,2)
      COMMON/CM15/VJXIX(3,3,233),VJYIX(3,3,233)
      COMMON/CM15A/VJXIY(3,3,233),VJYIY(3,3,233)
      COMMON/CM16/DU(156),DV(156),ALFA(156),BETA(156),GAMMA(156)
      DIMENSION AVGPHI(156),DUMY(156),DUMY2(156,3)
C
      CALL RHSO
C
C     +DT*(<PHI(N)J.VJX,VIX>+<PHI(N)J.VJY,VIY>)
      DO 10 NODE = 1, NONODE
      AVGPHI(NODE) =((PHI(NODE,NP1)+PHI(NODE,NM1))/2.0)
      IF (IMATZU.EQ.1) AVGPHI(NODE) = PHI(NODE,N)
   10 CONTINUE
      SIGN =  DT
      CALL TERM2 (AVGPHI,DUMY2,1,NM1,VJXIX,SIGN,1)
      CALL TERM2 (AVGPHI,DUMY2,1,NM1,VJYIY,SIGN,2)
C
C     +DT*<DVJ.VJX,VI>
      SIGN = DT
      CALL TERM4 (DV,VJX,SIGN,1)
C
C     -DT*<DUJ.VJY,VI>
      SIGN = -DT
      CALL TERM4 (DU,VJY,SIGN,2)
C
C     +DT*(<AKEJ.VJX,VIX>+<AKEJ.VJY,VIY>)
      SIGN =  DT
      CALL TERM2 (AKE,DUMY2,1,N,VJXIX,SIGN,1)
      CALL TERM2 (AKE,DUMY2,1,N,VJYIY,SIGN,2)
C
C     +<DIV(N-1)J.VJ,VI>
      SIGN = 1.0
      CALL TERM3 (DUMY,DIV,2,NM1,SIGN)
C
C     -DIVDIF*DT(<DIV.VJX,VIX>+<DIV.VJY,VIY>)
      SIGN =  -DIVDIF*DT
      CALL TERM2 (DUMY,DIV,2,NM1,VJXIX,SIGN,1)
      CALL TERM2 (DUMY,DIV,2,NM1,VJYIY,SIGN,2)
C
      DO 20 NODE = 1, NONODE
   20 SAVE(NODE,2) = RHS(NODE)
C
      RETURN
      END
```

```
      SUBROUTINE LEAPER (DT)
C
C     LEAPER - A SIMPLE MARCHING SCHEME WHERE F(N+1) USES TWO
C     PREVIOUS STEPS: F(N+1) = F(N-1) -2DT*F(N). THIS SCHEME IS
C     [illegible] COMPLETED. THE
C     [illegible]
C     [illegible]
C     [illegible] STEPS OF THREE HOURS EACH.
C
C     INPUT ARGUMENT          DT - DELTA T, TIME INTERVAL
C
      REAL*4 RUNNOR
      COMMON/CM1/NNODE,NELMT,NN,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/CM11/U(156),V(156),VOR(156),PSI(156),CHI(156)
      COMMON/CM12/ZETA(156,3),DIV(156,3),PHI(156,3),SAVE(156,2)
      COMMON/CM13/H(1044),G(1044),F(1044)
      COMMON/IPRINT/IPRNT(20),NAVE,R1,R2,LEAPHR,RUNNOR
      DIMENSION PHINEW(156)
C
C     INITIALIZE THE TIME LEVELS
      NM1 = 1
      N = 2
      NP1 = 3
      XHOUR = DT / 3600.0
C
      TWODT = 2.0*DT
C
C     CONSTRUCT MASS MATRIX NEEDED FOR CONTEQ & DIVEQ
      CALL AMTRX2 (TWODT)
C
C     LEAPFROG TO LEAPHR HOURS
      DO 30 ISTEP = 1, LEAPHR
C
      HOUR = HOUR + DT/3600.0
C
      CALL VORTEQ (TWODT)
      CALL RELAX (PSI,H,2)
C
      CALL CONTEQ (TWODT)
      CALL RELAX1 (PHINEW,F,1)
      DO 10 NODE = 1, NNODE
   10 PHI(NODE,NP1) = PHINEW(NODE)
C
      CALL DIVEQ  (TWODT,0)
      CALL RELAX (CHI,H,3)
C
      CALL DEPVAR
C
C     TEST FOR OUTPUT CONDITIONS
      XHOUR = XHOUR + DT/3600.0
      IF (IPRNT(12).GT.IFIX(XHOUR)) GO TO 20
      IF (IPRNT(11).EQ.1) CALL OUTPUT (7)
      IF (IPRNT(13).EQ.1) CALL OUTPUT (3)
      IF (IPRNT(14).EQ.1) CALL OUTPUT (8)
      IF (IPRNT(15).EQ.1) CALL PLOTSL
      IF (IPRNT(17).EQ.1) CALL HANAL
      XHOUR = XHOUR - IPRNT(12) + .1
C
C     UPDATE THE TIME LEVEL POINTERS
   20 NNM1 = NM1
      NM1 = N
      N = NP1
   30 NP1 = NNM1
C
      RETURN
      END
```

```
      SUBROUTINE TERM1 (ITERM,DUMY1,DUMY2,NDIMEN,NL)
C
C     TERM1 - CALCULATES THE TERMS IN THE RHS FOR SEVERAL
C     EQUATIONS ALPHA, BETA, AND ZE. TERM1 IS USED IN THE
C     COMPUTATION OF THE FIRST MULTIPLICATION OF THE DEPENDENT
C     VARIABLE IN A TRIPLE PRODUCT INTEGRATION. I.E. WHERE
C     U**2 = <UJ.VJ,UK.VK,VI>. A 3 BY 3 MATRIX BB IS
C     USED IN THE SAME WAY AS USED IN SUBROUTINE MATRX.
C     FINALLY THE TERM IS ASSEMBLED INTO THE WORKING VECTOR
C     CALLED C, BY ASEMBL. NOTE IN COMMON STATEMENT /CM10/
C     THE SECOND ARGUMENT IS NAMED P. IN FACT THIS IS THE VIJK
C     INNER PRODUCT THAT IS USED IN THIS SUBROUTINE. BUT FOR
C     COMPACTION OF CODE PER LINE IT WAS RENAMED TO P.
C
C     INPUT ARGUMENTS    ITERM - FLAG USED FOR BOUNDARY TEST
C                        DUMY1 - DEPENDENT VARIABLE (156)
C                        DUMY2 - DEPENDENT VARIABLE (156,3)
C                        NDIMEN - FLAG FOR WHICH DEPENDENT
C                                 VARIABLE IS TO BE USED
C                        NL - CURRENT TIME LEVEL FOR DUMY2
C
      INTEGER*2 ELMENT,IBD
      COMMON/CM1/NONODE,NOELMT,NN,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/CM2/ELMENT(283,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/CM5/IBD(156),NOPT,NTRI(156),XI(12),YI(13)
      COMMON/CM10/VIJ(283),P(3,3,3,283),VIJY(3,3,3,283)
      COMMON/CM14/BB(3,3),C(1044)
      DIMENSION DUMY1(156),DUMY2(156,3)
C
      DO 100 I=1,NOELMT
      N1=ELMENT(I,1)
      N2=ELMENT(I,2)
      N3=ELMENT(I,3)
C
      GO TO (10,20), NDIMEN
   10 D1=DUMY1(N1)
      D2=DUMY1(N2)
      D3=DUMY1(N3)
      GO TO 30
   20 D1 = DUMY2(N1,NL)
      D2 = DUMY2(N2,NL)
      D3 = DUMY2(N3,NL)
C
C     CORRECT FOR BOUNDARY WHEN ITERM = 2
   30 GO TO (50,40),ITERM
   40 IF (IBD(N1).EQ.5) D1=0.0
      IF (IBD(N2).EQ.5) D2=0.0
      IF (IBD(N3).EQ.5) D3=0.0
C
C     COMPUTE INTERACTION BETWEEN NODES; PASS BB TO ASEMBL
   50 BB(1,1) = D1*P(1,1,1,I) +D2*P(2,1,1,I) +D3*P(3,1,1,I)
      BB(1,2) = D1*P(1,2,1,I) +D2*P(2,2,1,I) +D3*P(3,2,1,I)
      BB(1,3) = D1*P(1,3,1,I) +D2*P(2,3,1,I) +D3*P(3,3,1,I)
      BB(2,1) = D1*P(1,1,2,I) +D2*P(2,1,2,I) +D3*P(3,1,2,I)
      BB(2,2) = D1*P(1,2,2,I) +D2*P(2,2,2,I) +D3*P(3,2,2,I)
      BB(2,3) = D1*P(1,3,2,I) +D2*P(2,3,2,I) +D3*P(3,3,2,I)
      BB(3,1) = D1*P(1,1,3,I) +D2*P(2,1,3,I) +D3*P(3,1,3,I)
      BB(3,2) = D1*P(1,2,3,I) +D2*P(2,2,3,I) +D3*P(3,2,3,I)
      BB(3,3) = D1*P(1,3,3,I) +D2*P(2,3,3,I) +D3*P(3,3,3,I)
      CALL ASEMBL (I,C)
  100 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE TERM2 (DUMY1,DUMY2,NDIMEN,NL,P,SIGN,IBDRY)
C
C     TERM2 - COMPUTES THE AREA INTEGRATION OVER EACH ELEMENT
C     FOR TERMS INVOLVING THE INNER PRODUCT BETWEEN THE BASIS
C     AND TEST FUNCTIONS ARE THEIR DERIVATIVES, I.E. <U,UX,ZIX>.
C
C     INPUT ARGUMENTS       DUMY1 - DEPENDENT VARIABLE (156)
C                           DUMY2 - DEPENDENT VARIABLE (156,3)
C                           NDIMEN - FLAG FOR WHICH DEPENDENT
C                                  VARIABLES IS USED
C                           NL - TIME LEVEL FOR DUMY2
C                           P - INNER PRODUCT
C                           SIGN - SIGN OF THE TERM
C                           IBDRY - FLAG USED FOR BOUNDARY
C
      INTEGER*2 ELMENT,IBD
      INTEGER ELMNBR
      COMMON/CM1/NNODE,NDELMT,DT,HDUM,PHISAR,NL,N,PI,IFLG
      COMMON/CM2/ELMENT(233,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/CM4/RHS(156),AMASSU(156),AMASSI(156),PACT(156,3)
      COMMON/CM5/IBD(156),IOPT,NTNI(156),XI(12),YI(13)
      DIMENSION DUMY1(156),DUMY2(156,3),P(3,3,233)
C
C     CALCULATE THE INTERACTION FOR EACH NODE PER ELEMENT
      DO 80 ELMNBR = 1, NDELMT
      NODE1 = ELMENT(ELMNBR,1)
      NODE2 = ELMENT(ELMNBR,2)
      NODE3 = ELMENT(ELMNBR,3)
C
      GO TO (20,40), NDIMEN
   20 D1 = DUMY1(NODE1)
      D2 = DUMY1(NODE2)
      D3 = DUMY1(NODE3)
      GO TO 50
   40 D1 = DUMY2(NODE1,NL)
      D2 = DUMY2(NODE2,NL)
      D3 = DUMY2(NODE3,NL)
C
C     COMPUTE INTERACTION BETWEEN NODES
   50 I = ELMNBR
      TEMP1 = D1*P(1,1,I) + D2*P(1,2,I) + D3*P(1,3,I)
      TEMP2 = D1*P(2,1,I) + D2*P(2,2,I) + D3*P(2,3,I)
      TEMP3 = D1*P(3,1,I) + D2*P(3,2,I) + D3*P(3,3,I)
C
C     CORRECT FOR BOUNDARY WHEN IBDRY = 2
      GO TO (70,60), IBDRY
   60 IF(IBD(NODE1).EQ.5) TEMP1 = 0.0
      IF(IBD(NODE2).EQ.5) TEMP2 = 0.0
      IF(IBD(NODE3).EQ.5) TEMP3 = 0.0
C
C     ADD RESULTS TO THE RHS (WORK VECTOR)
   70 RHS(NODE1) = RHS(NODE1) + TEMP1*SIGN
      RHS(NODE2) = RHS(NODE2) + TEMP2*SIGN
      RHS(NODE3) = RHS(NODE3) + TEMP3*SIGN
   80 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE TERM3 (DUMY1,DUMY2,NDIMEN,NL,S)
C
C  TERM3 - COMPUTES THE AREA INTEGRATION OVER EACH ELEMENT
C  FOR EACH NODE WHERE THE INNER PRODUCT BETWEEN THE BASIS
C  AND TEST FUNCTION ARE AS FOLLOWS, I.E. <VJ,VI>.
C
C     INPUT ARGUMENTS      DUMY1 - DEPENDENT VARIABLE (155)
C                          DUMY2 - DEPENDENT VARIABLE (155,3)
C                          NDIMEN - FLAG FOR WHICH DEPENDENT
C                                VARIABLES IS USED
C                          NL - TIME LEVEL FOR DUMY2
C                          SIGN - SIGN OF THE TERM
C
      INTEGER*2 NUM,ISTART,NAME
      COMMON/CM1/NNODE,NOELMT,NT,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/CM5/NUM(155),ISTART(155),NAME(1044)
      COMMON/CM4/RHS(155),NMASSJ(155),NMASSI(155),PAST(155,3)
      COMMON/CM13/A(1044),G(1044),F(1044)
      DIMENSION DUMY1(155),DUMY2(155,3)
C
      GO TO (20,60), NDIMEN
C
C
   20 DO 40 NODE = 1, NONODE
      IFIRST = ISTART(NODE)
      LAST = IFIRST + NUM(NODE) - 1
      DO 40 INDEX = IFIRST, LAST
      RHS(NODE) = RHS(NODE) + DUMY1(NAME(INDEX))*G(INDEX)*S
   40 CONTINUE
      GO TO 100
C
C
   60 DO 80 NODE = 1, NONODE
      IFIRST = ISTART(NODE)
      LAST = IFIRST + NUM(NODE) - 1
      DO 80 INDEX = IFIRST, LAST
      RHS(NODE) = RHS(NODE) + DUMY2(NAME(INDEX),NL)*G(INDEX)*S
   80 CONTINUE
C
  100 RETURN
      END
```

```
      SUBROUTINE TERM4 (D,P,SIGN,IBDRY)
C
C   TERM4 - COMPUTES THE ... INTEGRATION OVER EACH ELEMENT
C   FOR EACH NODE WHERE THE INNER PRODUCT BETWEEN THE BASIS
C   AND TEST FUNCTIONS ARE NS FIELDS, I.E. <VSK,VI>.
C
C     INPUT VARIABLES        D - DEPENDENT VARIABLE
C                            P - INNER PRODUCT
C                            SIGN - SIGN OF THE TERM
C                            IBDRY - FLAG USED FOR BOUNDARY
C
      INTEGER*2 ELMENT,NUM,ISTART,NAME,IBD
      INTEGER ELMNBR
      COMMON/C11/NNODE,NOELMT,NN,NOUR,PHIBAR,PHI,I,IP1,IFLG
      COMMON/C12/ELMENT(255,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/C14/RHS(156),AMASSJ(156),AMASSI(156),PAST(156,3)
      COMMON/C15/IBD(156),IOPT,NTRI(156),X1(12),Y1(13)
      DIMENSION D(156), P(3,255)
C
C     CALCULATE THE INTERACTION FOR EACH NODE PER ELEMENT
      DO 30 ELMNBR = 1, NOELMT
      NODE1 = ELMENT(ELMNBR,1)
      NODE2 = ELMENT(ELMNBR,2)
      NODE3 = ELMENT(ELMNBR,3)
      D1 = D(NODE1)
      D2 = D(NODE2)
      D3 = D(NODE3)
C
C     COMPUTE TERM CONTRIBUTIONS
      I = ELMNBR
      TEMP1 = (D1*P(1,I) + D2*P(2,I) + D3*P(3,I))*SIGN
      TEMP2 = (D1*P(1,I) + D2*P(2,I) + D3*P(3,I))*SIGN
      TEMP3 = (D1*P(1,I) + D2*P(2,I) + D3*P(3,I))*SIGN
C
C     CORRECT FOR BOUNDARY WHEN IBDRY = 2
      GO TO (20,10), IBDRY
C  10 CONTINUE
   10 IF(IBD(NODE1).EQ.5) TEMP1 = 0.0
      IF(IBD(NODE2).EQ.5) TEMP2 = 0.0
      IF(IBD(NODE3).EQ.5) TEMP3 = 0.0
C
C     ADD RESULTS TO THE RHS (WORK VECTOR)
   20 RHS(NODE1) = RHS(NODE1) + TEMP1
      RHS(NODE2) = RHS(NODE2) + TEMP2
      RHS(NODE3) = RHS(NODE3) + TEMP3
   30 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE OUTPUT (IFROM)
C
C     OUTPUT - PRINTS THE CONTENTS OF ONE OR MORE OF THE ARRAYS
C     AND/OR RESULTS FROM WHICH IT IS CALLED.
C
C     INPUT ARGUMENT  IFROM - POINTER TO OUTPUT FORMATING
C
      REAL*8 BUFFER
      INTEGER*2 ELMENT,NUM,ISTART,NAME,IBD
      INTEGER ELMNBR,START
      COMMON/CM1/NNODE,NELMT,NN,NBOUN,PHIBAR,NCIL,M,NP1,IFLG
      COMMON/CM1A/THETA,THETAO,DELTAX,DELTAY,HLAT,NLONG,DT
      COMMON/CM2/ELMENT(233,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/CM3/NUM(156),ISTART(156),NAME(1044)
      COMMON/CM5/IBD(156),NOPT,NTRI(156),XI(12),YI(12)
      COMMON/CM6/THETAO,XL,H,VEL,YTOP,YBOT,YLAMDA,E,UBAR,FO
      COMMON/CM7/XLONG (156),YLAT(156),AMP,SMCOEF,SCR(156)
      COMMON/CM8/A(3,233),B(3,233),AREA2(233)
      COMMON/CM9/VJX(3,233),VJY(3,233),VIJX(3,3,3,233)
      COMMON/CM10/VIJ(233),VIJK(3,3,3,233),VIJY(3,3,3,233)
      COMMON/CM11/U(156),V(156),XKE(156),PSI(156),CHI(156)
      COMMON/CM12/ZETA(156,3),DIV(156,3),PHI(156,3),SAVE(156,2)
      COMMON/CM13/H(1044),G(1044),F(1044)
      COMMON/CM14/BB(3,3),C(1044)
      COMMON/CM15/VJKIX(3,3,233),VJYIX(3,3,233)
      COMMON/CM15A/VJKIY(3,3,233),VJYIY(3,3,233)
      COMMON/CM16/QU(156),QV(156),ALFA(156),BETA(156),UNM1(156)
      COMMON/CM17/XMIN,YMIN,FCSTHR
      COMMON/IPRINT/IPRNT(20),NAVEND,R1,R2,LEAPHR,RUNNBR
      DIMENSION II(12),ITEST(7)
C
      GO TO(10,30,40,50,60,100,70,80,90),IFROM
C
C
C
C     PRINT FOR EACH ELEMENT ALL OF ITS NODES  < CORRES >
C
   10 WRITE (6,11)
   11 FORMAT(///,10X,'ELEMENT #     NODES',/)
      DO 12 ELMNBR = 1, NELMT
   12 WRITE (6,13) ELMNBR, (ELMENT(ELMNBR,NODE),NODE=1,3)
   13 FORMAT(10X,I5,2X,3I6)
C
C
C
C
C     PRINT THE CONNECTIVITY MATRIX NAME  < CORREL >
C
      WRITE (6,21)
   21 FORMAT(///,10X,'THE CONNECTIVITY MATRIX LISTED ',
     1'BY NODE')
      WRITE(6,22)
   22 FORMAT(/,10X,'NODE    ISTART    NUM    CONNECTIVITY ',
     1'NODES')
      DO 24 NODE = 1, NNODE
      START = ISTART(NODE)
      L = NUM(NODE) + START - 1
      WRITE (6,23) NODE,START,NUM(NODE),(NAME(I),I=START,L)
   23 FORMAT(I13,I8,I7,2X,3I4)
   24 CONTINUE
      WRITE (6,25) NN
   25 FORMAT(///,10X,'TOTAL NUMBER OF CONNECTIVITY NODES ',
     1'=',I6)
      WRITE (6,26)
   26 FORMAT(/,10X,'CONNECTIVITY VECTOR NAME')
      WRITE (6,27) (NAME(K),K=1,NN)
   27 FORMAT (25I5)
      GO TO 1000
C
```

```
C
C
C     PRINT THE MASS MATRICES G, H AND F    < MTRX2 >
C
   30 WRITE(6,31)
   31 FORMAT(///,10X,'MASS MATRIX G     USED IN SOLVING THE',
     1' DEPENDENT VARIABLES',/)
      WRITE(6,33) (G(J),J=1,NN)
   33 FORMAT(8E15.6)
      WRITE(6,34)
   34 FORMAT(///,10X,'MASS MATRIX H USE TO COMPUTE VDOTTED',
     1' AND DIVE,',/)
      WRITE(6,33) (H(J),J=1,NN)
      WRITE(6,35)
   35 FORMAT(///,10X,'MASS MATRIX F USE TO COMPUTE CONTE.',/)
      WRITE(6,33) (F(J),J=1,NN)
      GO TO 10000
C
C
C     PRINT AREA2 ARRAY    <AREA>
C
   40 WRITE(6,41)
   41 FORMAT(///,10X,'AREA2 ARRAY - AREA''S OVER THE ',
     1'ELEMENTS',/)
      WRITE (6,43) (AREA2(I),I=1,50)
      WRITE(6,45)
   45 FORMAT(///,3X,'A(1,K)',8X,'A(2,K)',8X,'A(3,K)',8X,
     1'B(1,K)',8X,'B(2,K)',8X,'B(3,K)        WHERE K = 1, 50')
      DO 42 K=1,288
   42 WRITE (6,43) (A(I,K),I=1,3),(B(I,K),I=1,3),AREA2(K)
   43 FORMAT (6E14.6,10X,E14.6)
C
C
C
C     PRINT X, Y COORDINATES FOR THE GRID POINTS <LOCATE>
C
   50 WRITE(6,51)
   51 FORMAT(///,10X,'X COORDINATES',/)
      WRITE(6,52) (XLONG(I),I=1,NONODE)
   52 FORMAT(1X,12F10.1)
      WRITE(6,53)
   53 FORMAT(///,10X,'Y COORDINATES',/)
      WRITE(6,52) (YLAT(I),I=1,NONODE)
      WRITE (6,55)
   55 FORMAT (///,6X,'CHANNEL LENGTH',1X,'CHANNEL WIDTH',6X,
     1'YBOT',11X,'YBOT',10X,'DELTAY',9X,'DELTAX',9X,
     2'THETA0',4X,'CORIOLIS VALUE')
      WRITE (6,56) XL,W,YTOP,YBOT,DELTAY,DELTAX,THETA0,VEL
   56 FORMAT (3X,8E15.5)
      GO TO 10000
C
C
C     PRINT THE INNER PRODUCTS   <INNER>
C
   60 WRITE(6,61)
   61 FORMAT(///,10X,'INNER PRODUCT    VIJ = <VJ,VI>          ',
     1'  FIRST 43 ELEMENTS',/)
      WRITE(6,62) (VIJ(ELMNBR),ELMNBR=1,30)
   62 FORMAT(1X,6E15.5)
      WRITE(6,63)
   63 FORMAT(///,10X,'INNER PRODUCT   VIJK = <VJ,VK,VI>',/)
      DO 65 I=1,3
      WRITE(6,15) I
   15 FORMAT(/,10X,'ELEMENT   ',I5,'   ********************',/)
      DO 64 K=1,3
      DO 67 M=1,3
   67 WRITE(6,66) (VIJK(K,M,J,I),J=1,3)
   66 FORMAT(1X,3F15.4)
```

```
54 WRITE(6,57)
55 CONTINUE
57 FORMAT(/)
   WRITE(6,58)
49 FORMAT(///,10X,'INNER PRODUCT      VJK = <VJK,VI>      ',
  1...,'     INNER PRODUCT      VJY = <VJY,VI>      (40',
  2' ELEMENTS',/)
   DO 50 I=1,40
50 WRITE(6,57) (VJX(J,I),J=1,3),(VJY(JJ,I),JJ=1,3)
57 FORMAT(1X,3F15.4,10X,'*****',5X,3F15.4)
   WRITE(6,58)
58 FORMAT(///,10X,'INNER PRODUCT   VJXIK = <VJK,VIK>      ',
  117X,'INNER PRODUCT   VJYIY = <VJY,VIY>      43 ELEMENTS',
  2/)
   DO 59 I=1,43
   WRITE(6,43) I
43 FORMAT(I12)
   DO 49 K=1,3
49 WRITE(6,57) (VJXIX(K,J,I),J=1,3),(VJYIY(K,JJ,I),JJ=1,3)
59 CONTINUE
   GO TO 10000
C
C
C  PRINT FIELD PSI, CHI AND PHI    <IC>
C
C
70 WRITE(6,71) HOUR
71 FORMAT(///,10X,'PHI FIELD       TIME = ',F4.1,' HRS',/)
   WRITE (6,72) (PHI(I,NP1),I=1,NONODE)
72 FORMAT (1X,12E11.4)
   WRITE(6,73) HOUR
73 FORMAT(///,10X,'PSI FIELD       TIME = ',F4.1,' HRS',/)
   WRITE (6,75) ( PSI(I),I=1,NONODE)
   WRITE(6,74) HOUR
74 FORMAT(///,10X,'CHI FIELD       TIME = ',F4.1,' HRS',/)
   WRITE (6,75) ( CHI(I),I=1,NONODE)
75 FORMAT(1X,12E11.4)
   GO TO 10000
C
C
C  PRINT THE DEPENDENT VARIABLE FIELDS     <INITOP>
C
C
80 WRITE(6,81) HOUR
81 FORMAT(///,10X,'U FIELD          TIME = ',F4.1,' HRS',/)
   WRITE (6,82) (U(NODE),NODE=1,NONODE)
82 FORMAT(1X,12E11.4)
   WRITE(6,83) HOUR
83 FORMAT(///,10X,'V FIELD          TIME = ',F4.1,' HRS',/)
   WRITE (6,82) (V(NODE),NODE=1,NONODE)
   WRITE(6,84) HOUR
84 FORMAT(///,10X,'KE FIELD          TIME = ',F4.1,' HRS',/)
   WRITE (6,82) (XKE(NODE),NODE=1,NONODE)
   WRITE(6,85) HOUR
85 FORMAT(///,10X,'VORTICITY FIELD          TIME = ',F4.1,
  1' HRS',/)
   WRITE (6,82) (ZETA(NODE,NP1),NODE=1,NONODE)
   WRITE(6,86) HOUR
86 FORMAT(///,10X,'DIVERGENT FIELD          TIME = ',F4.1,
  1' HRS',/)
   WRITE (6,82) (DIV(NODE,NP1),NODE=1,NONODE)
   WRITE(6,87) HOUR
87 FORMAT(///,10X,'QU FIELD          TIME = ',F4.1,' HRS',/)
   WRITE (6,82) (QU(NODE),NODE=1,NONODE)
   WRITE(6,88) HOUR
88 FORMAT(///,10X,'QV FIELD          TIME = ',F4.1,' HRS',/)
   WRITE (6,82) (QV(NODE),NODE=1,NONODE)
   WRITE(6,89) HOUR
89 FORMAT(///,10X,'ALFA FIELD          TIME = ',F4.1,' HRS',
  1/)
   WRITE (6,82) (ALFA(NODE),NODE=1,NONODE)
```

```
      WRITE(6,79) HOUR
   79 FORMAT(///,10X,'BETA FIELD          TIME = ',F4.1,' HRS',
     */)
      WRITE (6,92) (BETA(NODE),NODE=1,NONODE)
      GO TO 1000
C
C
C     PRINT THE DEPENDENT VARIABLE FIELDS U AND V    <LEAPFR>
C
   90 WRITE(6,91) HOUR
   91 FORMAT(///,10X,'U FIELD          TIME = ',F4.1,' HRS',/)
      WRITE (6,92) (U(NODE),NODE=1,NONODE)
   92 FORMAT(1X,10E11.4)
      WRITE(6,93) HOUR
   93 FORMAT(///,10X,'V FIELD          TIME = ',F4.1,' HRS',/)
      WRITE (6,92) (V(NODE),NODE=1,NONODE)
C
 1000 RETURN
      END
```

```
      SUBROUTINE PLOTS1
C
C PLOTS1 - SETS UP ALL THE NECESSARY ARRAYS TO BE PASSED TO
C PLOTS3, AND DOES ALL THE GRAPHICS ON THE VERSATEC PLOTTER.
C
      REAL*8 TD
      COMMON/CNST/ ... E, ... T, ..., PHIDSF, ... 1, ..., NP1,IFLD
      COMMON /CNTL/THETA,THETAO,DELTAX,DELTAY,ALAT,NLONG,DT
      COMMON/CMPL1/U(155),V(155),W(155),PSI(155),CHI(155)
      COMMON/CMPL2/ZETA(155,3),DIV(155,3),PHI(155,3),SAVE(155,2)
      COMMON /IPRINT/IPRINT(20),NAVE,J,K1,K2,LENPAR,KOUNT
      DIMENSION IB(43),NB(11),NA(10),CL(12)
      DIMENSION IWA(168),WORKX(155),WORKY(155),ITEST(7)
      REAL*8 TITLE(12),TITLE1(7),TITLE2(14)
      DATA TITLE/'        ','        ','        ',' HOUR   ','  PROG  ',
     1'        ','        ',' RUN    ','        ','        ','        ',
     2'        ','        '/
      DATA TITLE1/'   PHI  ','   PSI  ','   CHI  ',
     1'   U    ','   V    ','  VORT  ','  DIV   '/
      DATA TITLE2/'         0','       1.5','         3',
     1'         6','         9','        12','        15',
     2'        18','        21','        24','        30',
     3'        36','        42','        48'/
      DATA NA/1,1,0,0,1,0,0,0,0,0/
C
      DATA ITEST/1000000,100000,10000,1000,100,10,1/
C
      IER = 0
      IPLOT = IPRINT(15)
      NBDRY = 30
      NPTS = 163
      CALL PLOTS2 (WORKX,WORKY,NPTS,IB,NBDRY)
      TITLE(3) = NUMBR
C
      DO 100 INDEX = 1, 7
      IF (IPLOT.LT.ITEST(INDEX)) GO TO 100
C
      GO TO (10,20,30,40,50,60,70), INDEX
C
   10 CALL PLOTS3 (DUMY,PHI,2,NP1,WORK,NPTS,1.0,WMAX,WMIN)
      GO TO 80
C
   20 CALL PLOTS3 (PSI,DUMY,1,NP1,WORK,NPTS,0.0001,WMAX,WMIN)
      GO TO 80
C
   30 CALL PLOTS3 (CHI,DUMY,1,NP1,WORK,NPTS,1.0,WMAX,WMIN)
      GO TO 80
C
   40 CALL PLOTS3 (U,DUMY,1,NP1,WORK,NPTS,1.0,WMAX,WMIN)
      GO TO 80
C
   50 CALL PLOTS3 (V,DUMY,1,NP1,WORK,NPTS,1.0,WMAX,WMIN)
      GO TO 80
C
   60 CALL PLOTS3 (DUMY,ZETA,2,NP1,WORK,NPTS,0.1E+06,WMAX,WMIN)
      GO TO 80
C
   70 CALL PLOTS3 (DUMY,DIV,2,NP1,WORK,NPTS,1.0,WMAX,WMIN)
C
   80 DF = ABS(WMAX-WMIN)/11.0
      CL(1) = WMIN
      DO 85 I = 2, 12
   85 CL(I) = CL(I-1) + DF
      TITLE(1) = TITLE1(INDEX)
C
C
C
C
C
C
```

```
C
      IF(HOUR.LE. 0.1 ) TITLE(2) = TITLE2(1)
      IF(HOUR.GE.1.4)   TITLE(2) = TITLE2(2)
      IF(HOUR.GE.2.4)   TITLE(2) = TITLE2(3)
      IF(HOUR.GE.5.3)   TITLE(2) = TITLE2(4)
      IF(HOUR.GE.8.7)   TITLE(2) = TITLE2(5)
      IF(HOUR.GE.11.1)  TITLE(2) = TITLE2(6)
      IF(HOUR.GE.14.4)  TITLE(2) = TITLE2(7)
      IF(HOUR.GE.17.8)  TITLE(2) = TITLE2(8)
      IF(HOUR.GE.20.3)  TITLE(2) = TITLE2(9)
      IF(HOUR.GE.23.1)  TITLE(2) = TITLE2(10)
      IF(HOUR.GE.29.7)  TITLE(2) = TITLE2(11)
      IF(HOUR.GE.35.0)  TITLE(2) = TITLE2(12)
      IF(HOUR.GE.41.2)  TITLE(2) = TITLE2(13)
      IF(HOUR.GE.47.4)  TITLE(2) = TITLE2(14)
C
      YA(1) = IBDRY
      NC = 12
      CF = 1.0
      XS = -300000.0
      YS = -300000.0
C
      CALL CONISO (WORKX,WORKY,WORK,NPTS,I5,1,13,CL,NC,
     1CF,XS,YS,TITLE,XAX,YAX,YA,IER)
C
      IPLOT = IPLOT - ITEST(INDEX)
      WRITE(6,95) IER
   95 FORMAT(1X,'IER = ',I3)
  100 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE PLOTS2 (WORKX,WORKY,IDIMEN,IB,NBCRY)
C
C PLOTS2 - COMPUTES THE BOUNDARY POINTS AND FILLS THE
C [illegible] ARRAY TO BE PASSED TO CONIGO WITH THE FIELD.
C
      COMMON/CON1/ [illegible]ODE,[illegible]ELM,[illegible],RHO ,DELTAT, [illegible]1,[illegible],PI,IFLG
      COMMON /CON2/[illegible]/THETA[illegible],DELTAX,DELTAY, NLAT,NLONG,[illegible]
      COMMON /CON5/THETA),XL, ,YBL,YTOP,YBOT,XL[illegible],[illegible],FO
      COMMON /CON7/YLONG(150),YLAT(150),AZ,[illegible],SOR(150)
      DIMENSION WORKX(IDIMEN),WORKY(IDIMEN),IB(NBCRY)
C
C     COMPUTE X, Y COORD AND STORE
      IHFLAT = NLAT/2
      IHLTP1 = NLAT/2 + 1
      NT = NLAT + 1
      NODE1 = 0
      NODE = 0
      ICOD = 0
      DO 20 LAT = 1, NT
      DO 10 LONG = 1, NLONG
      NODE = NODE + 1
      NODE1 = NODE1 + 1
      WORKX(NODE) = XLONG(NODE1)
   10 WORKY(NODE) = YLAT(NODE1)
      IF (ICOD.EQ.1) GO TO 15
      ICOD = 1
      NODE = NODE + 1
C     WORKX(NODE) = XLONG(NODE1) + DELTAX
      WORKX(NODE) = XL
      WORKY(NODE) = YLAT(NODE1)
      GO TO 20
   15 ICOD = 0
   20 CONTINUE
C
C     BUILD THE IB BOUNDARY POINTS
      ICOUNT = 0
      NODE = -NLONG*2
      DO 30 INDEX = 1, IHLTP1
      ICOUNT = ICOUNT + 1
      NODE = NODE + NLONG*2 + 1
   30 IB(ICOUNT) = NODE
      DO 40 INDEX = 1,NLONG
      ICOUNT = ICOUNT + 1
      NODE = NODE + 1
   40 IB(ICOUNT) = NODE
      DO 50 INDEX = 1, IHFLAT
      ICOUNT = ICOUNT + 1
      NODE = NODE - (NLONG*2 + 1)
   50 IB(ICOUNT) = NODE
      NT = NLONG - 1
      DO 60 INDEX = 1, NT
      ICOUNT = ICOUNT + 1
      NODE = NODE - 1
   60 IB(ICOUNT) = NODE
C
      RETURN
      END
```

```
      SUBROUTINE PLOTS3 (D1,D2,ITYPE,L,WORK,IDIMEN,F,WMAX,WMIN)
C
C     PLOTS3 - TRANSFERS THE FIELD TO BE PLOTTED TO THE
C     WORK ARRAY.
C
      COMMON /CMN1/ [illegible]
      COMMON/CMN2/ [illegible] ,DELTAY,NLAT,NLONG,DT
      COMMON/CMN3/XLONG(156),YLAT(156),AZP,[illegible],SCR(156)
      DIMENSION D1(156),D2(156,3),WORK(IDIMEN)
C
      NT = NLAT + 1
C
      GO TO (10,50), ITYPE
C
C     STORE IN VECTOR FORM
   10 WMAX = D1(1)*F
      WMIN = D1(1)*F
      NODE1 = 0
      NODE = 0
      IODD = 0
      DO 30 LAT = 1, NT
      DO 20 LONG = 1, NLONG
      NODE = NODE + 1
      NODE1 = NODE1 + 1
      WORK(NODE) = D1(NODE1) * F
      IF (WORK(NODE).GT.WMAX) WMAX = WORK(NODE)
      IF (WORK(NODE).LT.WMIN) WMIN = WORK(NODE)
   20 CONTINUE
      IF (IODD.EQ.1) GO TO 25
      IODD = 1
      NODE = NODE + 1
      NODE2 = NODE1 - NLONG + 1
      WORK(NODE) = D1(NODE2) * F
      IF (WORK(NODE).GT.WMAX) WMAX = WORK(NODE)
      IF (WORK(NODE).LT.WMIN) WMIN = WORK(NODE)
      GO TO 30
   25 IODD = 0
   30 CONTINUE
      GO TO 80
C
C     STORE MULTI DIMENSIONED ARRAY
   50 WMAX = D2(1,L)*F
      WMIN = D2(1,L)*F
      NODE1 = 0
      NODE = 0
      IODD = 0
      DO 70 LAT = 1, NT
      DO 60 LONG = 1, NLONG
      NODE = NODE + 1
      NODE1 = NODE1 + 1
      WORK(NODE) = D2(NODE1,L) * F
      IF (WORK(NODE).GT.WMAX) WMAX = WORK(NODE)
      IF (WORK(NODE).LT.WMIN) WMIN = WORK(NODE)
   60 CONTINUE
      IF (IODD.EQ.1) GO TO 65
      IODD = 1
      NODE = NODE + 1
      NODE2 = NODE1 - NLONG + 1
      WORK(NODE) = D2(NODE2,L) * F
      IF (WORK(NODE).GT.WMAX) WMAX = WORK(NODE)
      IF (WORK(NODE).LT.WMIN) WMIN = WORK(NODE)
      GO TO 70
   65 IODD = 0
   70 CONTINUE
C
   80 RETURN
      END
```

```
      SUBROUTINE HANAL
C
C     HANAL - SETS UP THE HARMONIC ANALYSIS, DONE BY HARMAN. EACH
C     ROW IS DONE INDEPENDENTLY. INTERP IS CALLED TO INTERPOLATE
C     THE VARIABLE RESOLUTION FIELDS TO A UNIFORM GRID.
C
      INTEGER*2 IBD
      REAL*8 RUNNBR
      COMMON/C11/NONODE,DELAT,IT,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/C11A/THETA,THETAU,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/C45/IBD(155),NOPT,NTRI(155),X1(12),Y1(13)
      COMMON/C111/U(155),V(155),WKB(155),PSI(155),CHI(155)
      COMMON/C112/ZETA(155,3),DIV(155,3),PHI(155,3),SAVE(155,2)
      COMMON/C116/UU(155),VV(155),ALFA(155),BETA(155),GAMI(155)
      COMMON/IPRINT/IPRNT(20),NAVEIB,R1,R2,LEAPHR,RUNNBR
      DIMENSION ITEST(7),FIELD(156),W(12)
C
      DATA ITEST/1000000,100000,10000,1000,100,10,1/
      IANAL = IPRNT(13)
C
      DO 100 INDEX = 1, 7
      IF (IANAL.LT.ITEST(INDEX)) GO TO 100
C
      GO TO (10,20,30,40,50,60,70), INDEX
C
   10 DO 15 NODE = 1, NONODE
   15 FIELD(NODE) = PHI(NODE,NP1)
      GO TO 80
C
   20 DO 25 NODE = 1, NONODE
   25 FIELD(NODE) = PSI(NODE)
      GO TO 80
C
   30 DO 35 NODE = 1, NONODE
   35 FIELD(NODE) = CHI(NODE)
      GO TO 80
C
   40 DO 45 NODE = 1, NONODE
   45 FIELD(NODE) = U(NODE)
      GO TO 80
C
   50 DO 55 NODE = 1, NONODE
   55 FIELD(NODE) = V(NODE)
      GO TO 80
C
   60 DO 65 NODE = 1, NONODE
   65 FIELD(NODE) = ZETA(NODE,NP1)
      GO TO 80
C
   70 DO 75 NODE = 1, NONODE
   75 FIELD(NODE) = DIV(NODE,NP1)
C
   80 WRITE (6,81) RUNNBR, HOUR
   81 FORMAT('1',///,20X,A8,10X,'TIME =',F6.1,'  HOUR',/)
C
      NLP = NLAT + 1
      DO 95 J=1, NLP
      JM = J - 1
      DO 90 I=1, NLONG
      K = JM * NLONG + I
      II = I
   90 CALL INTERP (X1(I),Y1(J),NTRI(K),FIELD,W,II)
   95 CALL HARMAN (J,JM,W,INDEX)
      IANAL = IANAL -ITEST(INDEX)
  100 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE INTERP (XX,YY,JTRI,FIELD,W,I)
C
C     THIS SUBROUTINE INTERPOLATES A VARIABLE RESOLUTION FIELD D INTO
C     A SERIES OF EVENLY SPACED LATITUDE CIRCLES, W . AS INPUT, IT
C     NEEDS THE COORDINATES (XX,YY) OF THE POINT TO BE INTERPOLATED,
C     AND ALSO THE IDENTIFICATION NUMBER (JTRI) OF THE ELEMENT IN
C     WHICH THE POINT IS FOUND .
C
      INTEGER*2 ELEMENT,NUM,ISTART,NAME,ID
      COMMON/CM1/NDNODE,NDELMT,NN,NODR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/CM1A/THETA,THETAO,DELTAX,DELTAY,NLAT,NLONG,DT
      COMMON/CM2/ELEMENT(288,3),PHIDIF,ZTADIF,DIVDIF
      COMMON/CM7/XLONG(156),YLAT(156),AMP,SPCOEF,SOR(156)
      DIMENSION FIELD(156),W(12)
C
      XL = DELTAX*NLONG
      XH = (XL / 2.)
      II = ELEMENT(JTRI,1)
      JJ = ELEMENT(JTRI,2)
      KK = ELEMENT(JTRI,3)
      TA = FIELD(II) - FIELD(JJ)
      TB = XLONG(II) - XLONG(KK)
      IF (ABS( TB ).GT. XH )TB = TB + XL
      TC = FIELD(II) - FIELD(KK)
      TD = XLONG(II) - XLONG(JJ)
      TF = YLAT(II) - YLAT(JJ)
      TG = YLAT(II) - YLAT(KK)
      C = ( TA*TB - TC*TD ) / ( TB*TF - TG*TD )
      IF ( TD .EQ. 0.0 ) GO TO 5
      B = ( TA - C*TF ) / TD
      GO TO 6
    5 B = ( TC - C*TG ) / TB
    6 A = FIELD(II) - C*YLAT(II) - B*XLONG(II)
      W(I) = A + B*XX + C*YY
C     WRITE(6,100)I,JTRI,XX,YY,FIELD(II),FIELD(JJ),FIELD(KK),W(I),TB
      RETURN
  100 FORMAT(' ',2I4,7F10.3)
      END
```

```
      SUBROUTINE HARM1 (IM,J1,Y,IFIELD)
C
      COMMON/C11/NONDIME,NDELMT,NN,HOUR,PHIBAR,NM1,N,NP1,IFLG
      COMMON/C11A/THETA,THETAU,DELTAX,DELTAY,NLAT,NLONG,DT
      DIMENSION Y(12),FC(12,7),FS(12,7)
      DIMENSION AMPL(7),PHASE(7),AF(7),BF(7)
      REAL*8 FIELD(7),FLD
      DATA FIELD/'  PHI    ','  PSI    ','  CHI    ',
     *'   U     ','   V     ','  VORT   ','  DIV    '/
C
C
      GO TO (1,2,3,4,5,6,7), IFIELD
    1 FLD = FIELD(1)
      GO TO 9
    2 FLD = FIELD(2)
      GO TO 9
    3 FLD = FIELD(3)
      GO TO 9
    4 FLD = FIELD(4)
      GO TO 9
    5 FLD = FIELD(5)
      GO TO 9
    6 FLD = FIELD(6)
      GO TO 9
    7 FLD = FIELD(7)
C
    9 IM=NLONG
      JM=NLAT+1
      Z=DELTAX
      I1  =   IM
      IF (J1) 10,10,27
   10 J1 = J1 + 1
      M11 = (I1/2) + 1
      M12 = M11 - 1
      E1 = 6.2831853/FLOAT(I1)
      E2 = 1.0/FLOAT(M12)
C
      WRITE(6,19) FLD,I1,JM,Z,M12
   19 FORMAT(//,A8,'E-W GRID POINTS =',I3,5X,
     *'N-S GRID POINTS =',I3,5X,
     *'GRID INCREMENT=',E14.6,5X,
     *'HIGHEST WAVE NUMBER POSSIBLE=',I3,//)
C
      DO 20 I = 1,IM
      DO 20 J = 1,M11
      FC(I,J) = E2*COS(FLOAT((I   )*(J-1))*E1)
   20 FS(I,J) = E2*SIN(FLOAT((I   )*(J-1))*E1)
C
      DO 25 I = 1,IM
      FC(I,1) = 0.5*FC(I,1)
   25 FC(I,M11) = 0.5*FC(I,M11)
C
   27 M11 = (I1/2) + 1
      M12 = M11 - 1
      DO 40 J = 1,M11
      SUM = 0.0
      DO 30 I = 1,IM
   30 SUM = SUM + Y(I)*FC(I,J)
   40 AF(J) = SUM
C
C
      BF(1) = 0.0
      BF(M11) = 0.0
      DO 60 J = 2,M12
      SUM = 0.0
      DO 50 I = 1,IM
   50 SUM = SUM + Y(I)*FS(I,J)
   60 BF(J) = SUM
C
C
```

```
      DO 75 J = 1,N11
      PHASE(J) = 0.0
      IF (BF(J).EQ.0.0) GO TO 75
      IF ((BF(J).NE.0.0).AND.(AF(J).EQ.0.0)) GO TO 70
      PHASE(J) = ATAN(BF(J)/AF(J))
      GO TO 75
   70 PHASE(J) = 90.0*(BF(J)/ABS(BF(J)))
   75 CONTINUE
C
      DO 95 J = 1,N11
      IF (ABS(PHASE(J)) - 0.3) 80,80,90
   80 AMPL(J) = AF(J)/COS(PHASE(J))
      GO TO 95
   90 AMPL(J) = BF(J)/SIN(PHASE(J))
   95 CONTINUE
C
      DO 100 J = 1,N11
  100 PHASE(J) = 57.29578*PHASE(J)
C
      DO 120 J = 2,N11
      IF (AMPL(J)) 110,120,120
  110 IF (PHASE(J)) 112,112,115
  112 PHASE(J) = PHASE(J) + 180
      GO TO 118
  115 PHASE(J) = PHASE(J) - 180
  118 AMPL(J) = -AMPL(J)
  120 CONTINUE
C
      WRITE(6,121)
  121 FORMAT( /,' ',T5,'GRID PT',T20,'ARITHMETIC MEAN',T40,
     *T60,'PHASE'    )
      WRITE(6,122) NN,AMPL(1), ((AMPL(I), PHASE(I)),I=2,N11)
  122 FORMAT(T6,I2,T15,E11.4,5(T32,E11.4,12X,F8.2,/))
C
      RETURN
      END
```

LIST OF REFERENCES

Bathe, K. and Wilson, E.L., Numerical Methods In Finite Element Analysis, p. 71-123, Prentice-Hall, Inc., 1976.

Cullen, M.J.P., "A Simple Finite Element Method for Meteorological Problems," Journal Institute of Maths Applies, v. 11, p. 15 31, 1973.

______, "A Finite Element Method for a Non-Linear Initial Value Problem," Journal Institute of Math Apply, v. 13, p. 233-247, 1974.

______ and Hall, C.D., "Forecasting and General Circulation Results from Finite Element Models," Quarterly Journal of the Royal Meteorology Society, v. 105, p. 571-593, July 1979.

Haltiner, G.J. and Williams, R.T., Numerical Prediction and Dynamic Meteorology, 2nd ed., John Wiley and Sons, Inc., 1980.

Hinsman, D.E., Application of a Finite Element Method to the Barotropic Primitive Equations, M.S. Thesis, Naval Postgraduate School, Monterey, California, 116p., 1975.

Kelley, R.G., A Finite Element Prediction Model with Variable Element Size, M.S. Thesis, Naval Postgraduate School, Monterey, California, 109 p., 1976.

Norrie, D.E. and de Vries, G., The Finite Element Methods, Academic Press, 1973.

Older, M., A Two Dimensional Finite Element Advection Model with Variable Resolution, M.S. Thesis, Naval Postgraduate School, Monterey, California, 84 p., 1981.

Phillips, N.A., "Numerical Integration of the Primitive Equations on the Hemisphere," Monthly Weather Review, p. 333 345, September 1959.

Pinder, G.F. and Gray, W.G., Finite Element Simulation in Surface and Subsurface Hydrology, Academic Press, 1977.

Schoenstadt, A., " A Transfer Function Analysis of Numerical Schemes Used To Simulate Geostrophic Adjustment," Monthly Weather Review, v. 108, no. 8, p. 1248 1259, August 1980.

Shuman, F.G., "Numerical Weather Prediction," Bulletin American Meteorology Society, v. 59, no. 1, p. 5-17, January 1978.

Staniforth, A.N. and Mitchell, H.L., "A Semi-Implicit Finite Element Barotropic Model," Monthly Weather Review, v. 105, p. 154-169, February 1977.

Strang, G. and Fix, G.J., An Analysis of the Finite Element Method, Prentice Hall, Inc., 1973.

Williams, R.T., "On the Formulation of Finite Element Prediction Models," Monthly Weather Review, v. 109, no. 3, p. 463-466, March 1981.

_______ and Zienkiewicz, O.C., "Improved Finite Element Forms For The Shallow Water Equations," International Journal For Numerical Methods in Fluids, v. 1, no. 1, p. 81-97, January March 1981.

Zienkiewicz, O.C., The Finite Element Method in Engineering Science, McGraw Hill, 1971.

INITIAL DISTRIBUTION LIST

No. Copies

1. Defense Technical Information Center 2
   Cameron Station
   Alexandria, Virginia 22314

2. Library, Code 0142 2
   Naval Postgraduate School
   Monterey, California 93940

3. Dr. R.T. Williams, Code 63Wu 4
   Department of Meteorology
   Naval Postgraduate School
   Monterey, California 93940

4. Commander, Air Weather Service 1
   Military Airlift Command
   United States Air Force
   Scott Air Force Base, Illinois 62226

5. Captain Edward T. Woodward 2
   414 Felene Dr.
   Bellevue, Nebraska 68005

6. Captain Brian Van Orman 1
   AFIT/CIPF
   Wright-Patterson AFB, Ohio 45433

7. Dr. David A. Archer 1
   Douglas DuPont Rachford, Inc.
   6150 Chevy Chase
   Houston, Texas 77027

8. Prof. Gordon Bradley, Code 52Bz 1
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California 93940

9. Lt. P.J. Burger, Code 35 1
   Department of Meteorology
   Naval Postgraduate School
   Monterey, California 93940

10. Prof. Gilles Cantin, Code 69Ci 1
    Department of Mechanical Engineering
    Naval Postgraduate School
    Monterey, California 93940

11. Dr. M.J.P. Cullen 1
Meteorological Office
Bracknell, Berks, United Kingdom

12. Dr. R.L. Elsberry, Code 63Es 1
Department of Meteorology
Naval Postgraduate School
Monterey, California 93940

13. Dr. J.A. Galt 1
NOAA Pac Mar Envir Lab
University of Washington
Seattle, Washington 98105

14. Dr. R.L. Haney, Code 63Hy 1
Department of Meteorology
Naval Postgraduate School
Monterey, California 93940

15. LCDR Donald E. Hinsman, Code 35 1
Department of Meteorology
Naval Postgraduate School
Monterey, California 93940

16. Prof. Uno R. Kodres, Code 52Kr 1
Department of Computer Science
Naval Postgraduate School
Monterey, California 93940

17. Dr. Robert L. Lee 1
Atmospheric and Geophysical Science Division
University of California
P.O. Box 808
Livermore, California 94550

18. Prof. A.K. MacPherson 1
Lehigh University
Department of Mechanical Engineering
Bethleem, Pa 18015

19. Prof. C.N.K. Mooers, Code 68Mr 1
Chairman, Department of Oceanography
Naval Postgraduate School
Monterey, California 93940

20. Prof. R. Newton, Code 69Ne 1
Department of Mechanical Engineering
Naval Postgraduate School
Monterey, California 93940

21. Dr. N.A. Phillips 1
National Meteorological Center/NOAA
World Weather Building
Washington, D.C. 20233

22. Dr. T. Rosmond 1
Naval Environmental Prediction Research Facility
Monterey, California 93940

23. Prof. D. Salinas, Code 69Zc 1
Department of Mechanical Engineering
Naval Postgraduate School
Monterey, California 93940

24. Dr. Y. Sasaki 1
Department of Meteorology
University of Oklahoma
Norman, Oklahoma 73069

25. Prof. A.L. Schoenstadt, Code 53Zh 1
Department of Mathematics
Naval Postgraduate School
Monterey, California 93940

26. Dr. Andrew Staniforth 1
Recherce en Prevision Numerique
West Isle Office Tower, 5 ieme etage
2121 route Trans-Canada
Dorval, Quebec H9P1J3, Canada

27. Dr. W.C. Thacker 1
National Oceanic and Atmospheric Administration
15 Rickenbacker Causeway
Miami, Florida 33149

28. Dr. A. Weinstein 1
Naval Environmental Prediction Research Facility
Monterey, California 93940

29. Dr. F.J. Winninghoff, Code 35 1
Department of Meteorology
Naval Postgraduate School
Monterey, California 93940

30. Prof. O.C. Zienkiewicz 1
Head of Civil Engineering Department
Applied Science Building
Singleton Park
Swansea SA2 8PP
United Kingdom

Thesis 195572
W834 Woodard
c.1 Development of
improved finite element formulation for
shallow water equations.